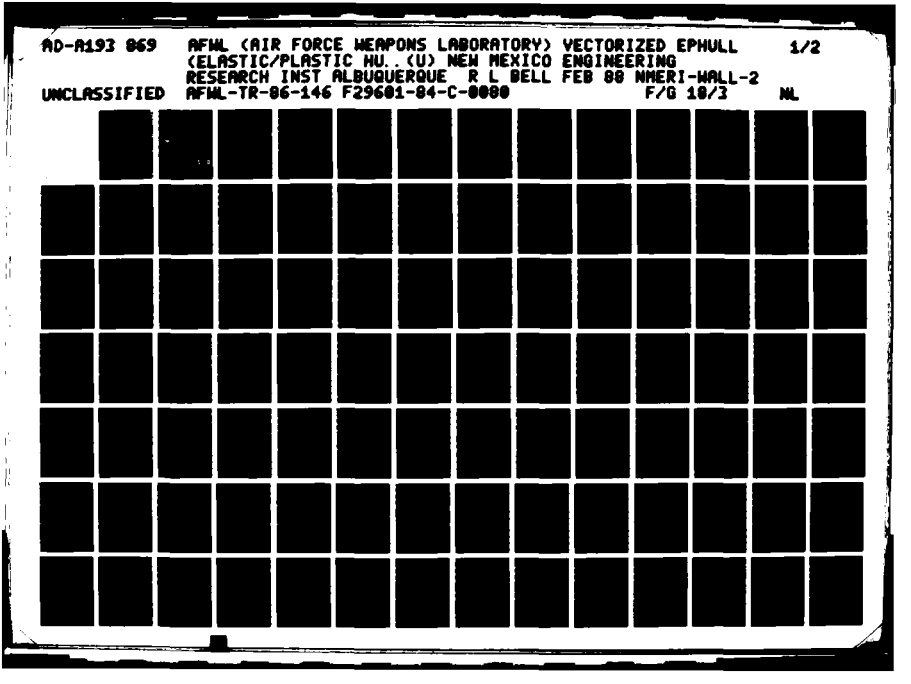
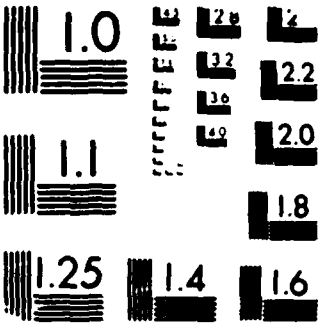

AD-A193 869 AFWL (AIR FORCE WEAPONS LABORATORY) VECTORIZED EPHULL 1/2
(ELASTIC/PLASTIC HU..(U) NEW MEXICO ENGINEERING
RESEARCH INST ALBUQUERQUE R L BELL FEB 88 NMERI-WALL-2
UNCLASSIFIED AFWL-TR-86-146 F29601-84-C-0080 F/G 10/3 NL



MICROCOPY RESOLUTION TEST CHART

AFWL-TR-
86-146

(2)

AD-A193 869

AFWL VECTORIZED EPHULL CODE USER MANUAL

R. L. Bell

New Mexico Engineering Research Institute
University of New Mexico
Albuquerque, NM 87131

February 1988

Final Report

Approved for public release; distribution unlimited.

AIR FORCE WEAPONS LABORATORY
Air Force Systems Command
Kirtland Air Force Base, NM 87117-6008

This final report was prepared by The New Mexico Engineering Research Institute, University of New Mexico, Albuquerque, New Mexico under Contract F29601-84-C-0080, Job Order 8809131C with the Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico. Capt Glenn E. James (NTEDA) was the Laboratory Project Officer-in-Charge.

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been authored by a contractor of the United States Government. Accordingly, the United States Government retains a nonexclusive, royalty-free license to publish or reproduce the material contained herein, or allow others to do so, for the United States Government purposes.

This report has been reviewed by the Public Affairs Office and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify AFWL/NTED, Kirtland Air Force Base, NM 87117-6008 to help us maintain a current mailing list.

This report has been reviewed and is approved for publication.

STEVEN D. WERT
Capt, USAF
Project Officer

FOR THE COMMANDER

FREDERICK M. JONAS
Lieutenant Colonel, USAF
Chief, Technology Br

CARL L. DAVIDSON
Colonel, USAF
Chief, Civil Engineering Research Div

DO NOT RETURN COPIES OF THIS REPORT UNLESS CONTRACTUAL OBLIGATIONS OR NOTICE ON A SPECIFIC DOCUMENT REQUIRES THAT IT BE RETURNED.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NMERI WA11-2	5 MONITORING ORGANIZATION REPORT NUMBER(S) AFWL-TR-86-146
6a NAME OF PERFORMING ORGANIZATION New Mexico Engineering Research Institute 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Air Force Weapons Laboratory
6c ADDRESS (City, State, and ZIP Code) PO Box 4825 Albuquerque, NM 87196	7b ADDRESS (City, State, and ZIP Code) Kirtland AFB, NM 87117-6008
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F29601-84-C-0080
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
62601F	8809	13	1C

11 TITLE (Include Security Classification)
AFWL VECTORIZED EPHULL CODE USER MANUAL

12 PERSONAL AUTHOR(S)
Bell, Raymond L.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM Mar 85 TO May 86	1988, February	154

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	05		HULL, Elastic-plastic, Hydrocode, Fluid dynamics, Equations-of-state, CTSS, BOW, Airblast prediction, Numerical modelling, EPHULL, Computer codes, Hydrodynamics, Finite difference (over

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This report was prepared as a user manual for implementing the vectorized Elastic/Plastic HULL (EPHULL) Composite HULL code and SAIL code on the AFWL CRAY computer. Major programs required to operate the Composite HULL code and supporting programs and files on the AFWL CRAY are described. The Composite HULL code developed through this work is usable over the range of problems formerly accommodated by either Vector HULL or EPHULL, and is simpler and less time-consuming for new users to learn. The Cylinder In Situ Test (CIST) Equation of State (EOS) in this manual was rewritten from the California Research Arbitrary Lagrangian-Eulerian (CRALE) code for use with a variety of soils.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
Capt Steven D. Wert	(505) 844-0261	AFWL/NTEDA

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

18. SUBJECT TERMS (continued)

CFS, COSMOS, Airblast code, High explosives predictions

PREFACE

This technical report has been prepared as a user manual for implementing the vectorized EPHULL (Composite HULL) code and SAIL code on the AFWL CRAY computer. Information in this report is organized into sections numbered for clarity and easy reference. Nonstandard terminology peculiar to computers is used as required to clearly and accurately convey essential information. Any engineering or scientific term given in other than SI units is shown in the form mandated for use in the relevant program(s). The symbol, case, and spacing of some mathematical terms are presented in required user and computer forms.

CONTENTS

CONTENTS (Concluded)

ILLUSTRATIONS

TABLES

TABLES (Continued)

TABLES (Concluded)

I. INTRODUCTION

1. OBJECTIVES AND REQUIREMENTS

New Mexico Engineering Research Institute (NMERI) was tasked to review the optimization methods used in the Vector HULL (sic) code, and to use those methods deemed adequate and effective, to produce a vectorized version of the latest available Elastic/Plastic HULL (EPHULL) code version. A further task was to implement the various boundary options available in Vector HULL but not in EPHULL, in the new version of EPHULL. The objective of this tasking was to create a new code (Composite HULL) able to handle all the different types of problems previously requiring the use of either Vector HULL or EPHULL. With the advent of Composite HULL, users will no longer have to learn or maintain multiple versions of the HULL code, thus reducing the time required to become a proficient code user. Also, user confusion created by the existence of several versions of HULL (each requiring different procedures) will be eliminated.

Another requirement was to rewrite the Cylinder In Situ Test (CIST) Equation Of State (EOS) from the California Research Arbitrary Lagrangian-Eulerian (CRALE) code (Ref. 1) for use in the Composite HULL code. This will give users a flexible equation of state, usable for various soils simply by changing the parameters in the material library as indicated by the in situ test results.

The deliverables include copies of all change files, copies of all COSMOS (job control language) job files, and a comprehensive final report. The final report was to include a description of all major programs required to operate the HULL system on the AFWL CRAY computer system.

2. REPORT DESCRIPTION

This report consists of seven sections and two appendixes. Sections I-IV narratively describe: (1) the effort objectives; (2) the U.S. Air Force Weapons Laboratory (AFWL) CRAY operating environment; (3) creation of the

---

1. Schuster, S., **Crale Users Manual**, AFWL-TR-82-45, Air Force Weapons Laboratory, Kirtland AFB, NM, September 1982.

Composite HULL code; and (4) the background and use of the preprocessor code SAIL. Section V describes the COSMOS job files for an example problem. Section VI summarizes the other modifications completed. Section VII provides conclusions and recommendations.

The appendixes are guides that can be used independently, for the SAIL code (Appendix A) and the Composite HULL code (Appendix B), as implemented on the AFWL CRAY computer. Appendix A contains a complete description of SAIL directives and procedures. Appendix B contains HULL system generation and operation procedures.

II. AFWL CRAY OPERATING ENVIRONMENT

1. CRAY TIME SHARING SYSTEM (CTSS) AND COSMOS

AFWL presently has a 2,000,000 decimal-word CRAY 1-S computer using the CRAY Time Sharing System (CTSS) operating system. This system was originally developed by the Lawrence Livermore National Laboratory (LLNL) and the Los Alamos National Laboratory (LANL). Under pressure from a small group of users, AFWL adopted CTSS in 1981 to replace the original CRAY Operating System (COS). The major "advantage" of CTSS over COS is that with CTSS, users can work interactively on the CRAY (COS was a batch-only system). Because of this interactive ability, CTSS does offer several capabilities not available under COS. One of the most useful of these capabilities is the Dynamic Debugging Tool (DDT), which allows a user to step through a large program to follow the progress of a calculation. DDT was used extensively in this project to detect subtle coding errors.

Production jobs can be submitted under CTSS either directly (from CTSS), or from a front-end machine. The job streams for production jobs use the COSMOS job control language. COSMOS commands are very similar to interactive CTSS commands. An asterisk (*) in the first column of a line must immediately precede a CTSS command in a COSMOS file (Ref. 2). COSMOS files may also be run interactively, by typing: COSMOS input file / time priority <carriage return>.

2. COMMON FILE SYSTEM (CFS)

Along with CTSS, AFWL also acquired the hardware and software to adopt the Common File System (CFS) developed by the national laboratories. CFS is a flexible permanent file system that uses an IBM mass store unit. Users must create root nodes under which files may be saved and subdirectories may be formed. The resulting tree structure allows files to be arranged in convenient patterns. The creator of a particular root node can give other users access by setting appropriate parameters. Files under a particular node may be made available for read-only by another user (or users), or read/write permission may be granted, etc. Figure 1 shows the CFS tree structure used by

---

2. **CTSS MINI-REFERENCE**, Los Alamos National Laboratory, August 1984.

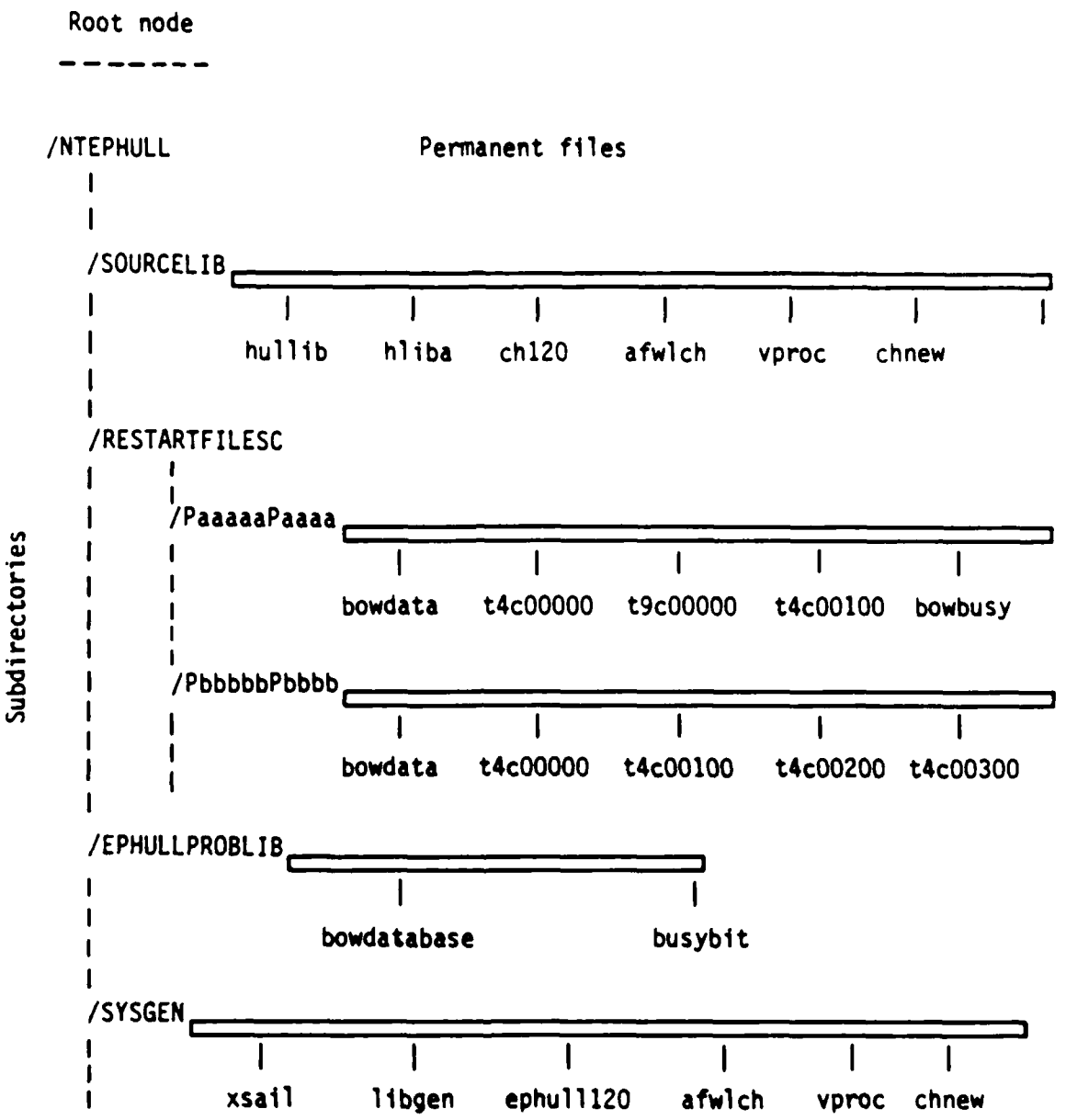


Figure 1. CFS Tree Structure; Root Node /NTEPHULL.

the HULL system (Ref. 3). All users are granted all rights under the HULL system root node /NTEPHULL.

When a file is obtained from CFS (by executing a MASS GET NAME command), the file is copied into the user's disk space. Unfortunately, no indication is given to CFS that the file obtained from CFS is in use. Thus, a file available to several users may be modified simultaneously by more than one user, then stored on CFS. Only the modifications added by the last user replacing the CFS file will be on that file; all other modifications will be lost.

When EPHULL was first modified for CTSS, Ms. Cydney Westmoreland of Orlando Technology, Inc.† (OTI) developed a unique method to allow the HULL system to maintain all problem numbers, in a database that avoids the multiple-user problem described above (Ref. 4). On CFS, a file that does not presently exist can be created by using either the MASS SAVE NAME or MASS STORE NAME command. If the file does exist, the MASS STORE NAME command causes the file NAME to be overwritten, but the MASS SAVE NAME command causes an error condition. Using Ms. Westmoreland's method, the HULL system program BOW attempts to SAVE a file (named BUSYBIT) under the node where the database file is maintained. If BUSYBIT is successfully saved, the database is accessed, modified, and stored. The BUSYBIT file is then deleted, allowing another user to access the database. If an error is detected when MASS SAVE BUSYBIT is attempted, the program detecting the error suspends itself for a time, then tries again. This process is repeated until the other program (the program that has saved BUSYBIT) completes its modifications and deletes BUSYBIT. A program that suspends itself an excessive number of times writes a message suggesting that BUSYBIT should be checked, and then aborts itself.

---

3. Bell, R. L., **Cray Time Sharing System (CTSS) Version of the HULL File Maintenance Program -- BOW**, AFWL-TR-84-58, Air Force Weapons Laboratory, Kirtland AFB, NM, August 1984.

4. Bell, R. and Westmoreland, C., **Elastic/Plastic HULL (EPHULL) Operation on the CRAY Time Sharing System (CTSS)**, AFWL-TR-83-6, Air Force Weapons Laboratory, Kirtland AFB, NM, April 1983.

† 60 Second St., Bldg. 5, Shalimar, FL 32579

III. COMPOSITE HULL CODE

1. HULL119 - THE PRIMARY SOURCE

The HULL family of hydrodynamics codes has been in existence for nearly 20 years (Ref. 5). Initially, all users were periodically given copies of the most recent "authorized" version. Eventually, however, several different versions of the code began evolving independently at different locations. An elastic/plastic version of the code (EPHULL) rapidly evolved at the Air Force Armament Laboratory (AFATL), Eglin AFB, FL. Other versions maintained by AFWL, the U. S. Army Ballistic Research Laboratory (BRL), and McDonnell Aircraft Co., continued to evolve as pure fluid codes (no stress formulation) used for air-blast and high-explosive (HE) simulation problems. The AFWL version of the code migrated to S-Cubed† when Mr. Charles Needham left AFWL. During the early 1980s, Mr. Daniel Matuska and Mr. John Osborn (of OTI) added a separate Lagrangian capability to EPHULL for AFATL. The Lagrangian module can be used independently, or in conjunction with the original Lagrangian-Eulerian remap scheme (Ref. 6).

When the composite HULL effort began, version 119 (HULL119) was the latest version of EPHULL. This code was used as the primary source code for composite HULL. It was decided to vectorize only the "number crunching" routines; i.e., those associated with the equations of state (EOS), hydro, and flux. Some minor changes were required in a few other routines, to maintain compatibility with the expanded EOS common blocks. An effort was made to maintain the order and names of the major subroutines, and the variable names and uses within these subroutines. Maximum resemblance to the scalar (EPHULL) code was thus retained, allowing people familiar with the scalar version to easily learn the vectorized version.

Although the vector changes were originally written for EPHULL119, at least two subsequent versions were produced and distributed since that time.

---

5. Durrett, R. E. and Matuska, D. A., **The HULL Code Finite Difference Solution to the Equations of Continuum Mechanics**, AFATL-TR-78-125, Air Force Armament Laboratory, Eglin AFB, FL, November 1978.

6. Matuska, D. A., and Osborn, J., **HULL Documentation**, Orlando Technology, Inc., 60 Second St., Bldg. 5, Shalimar, FL.

† Div. of Maxwell Laboratories, Inc., 5905 Marble N.E., Albuquerque, NM.

HULL119 was maintained as the base code until all vector routines were fully debugged, to eliminate the need to continuously update the sequence numbers on the change files each time a new version was distributed.

2. OTHER HULL VARIANTS

a. Vector HULL--In the early 1980s, the AFWL airblast version of HULL was completely rewritten by Dr. John Hasdal, to take advantage of the exceptional speed available through vector techniques on the CRAY computer (Ref. 7). His efforts produced an efficient new code, known as Vector HULL. Because Vector HULL does not use subroutine and variable names used in other versions of HULL, there is no direct correlation between Vector HULL and other versions.

Vector HULL abandoned the SAIL code for the commercially available UPDATE system, plus a preprocessor called POST. Vector HULL was divided into three different UPDATE libraries. Each library is made up of numerous UPDATE "decks." When making modifications, it may be difficult to determine which library contains the deck requiring change.

The Vector HULL version of program BOW uses a "lock file" scheme similar to EPHULL's BUSYBIT. The database is a buffered file. If it requires correction, the file must be decoded. Also, information concerning all files for all problems resides on this one file, so the various problems run under Vector HULL frequently cause saving and deleting of the lock file. A BOW program (Ref. 3) which uses ASCII files, has been written for EPHULL.

The methodologies used in Vector HULL have been carefully studied. Those considered most efficient have been incorporated into the vector routines of the new Composite HULL code.

Vector HULL also has a variety of boundary input capabilities based on the 1-kt standard (Ref. 8), and on station values from a previous calculation. These capabilities have also been implemented in the new code.

---

7. Vector HULL Report by J. Hasdal (unpublished, periodic progress report for Air Force Weapons Laboratory).

8. Needham, C. E., Havens, M. L., and Knauth, C. S., **Nuclear Blast Standard (1 kt)**, AFWL-TR-73-55, Air Force Weapons Laboratory, Kirtland Air Force Base, NM, April 1975.

b. Scalar HULL--This code is the most recent version of the original airblast code (Ref. 9). It was written when computer central memory was rather limited. Many of the routines available in Scalar HULL have proven quite useful to the airblast community. Most of these routines were previously transferred to the Vector HULL code, but both codes were used as sources for the 1-kt boundary input and the dust capabilities written for Composite HULL.

3. COMPOSITE HULL

a. Description--Composite HULL is the name given to the AFWL code incorporating the latest version of EPHULL, and the changes from files CHAFWL and CHVECT. These two change files have been modified, and the blocks of changes have been put in variously named "PROCs" (Appendix A). These PROCs are all added at a particular line sequence number, depending on the HULL version. A new file (AFWLCH) consists of a series of

```
*KEEPTO  LABEL  NOT "DEF" VECTOR
```

and

```
*INCLUDE  PROCNAME  "DEF" VECTOR
```

lines added at appropriate line sequence numbers for the latest version of EPHULL.

This last file (AFWLCH) has been transferred to AFATL, and may be included in the next update to EPHULL. Using the vector version of this code at AFWL will require only the current version of EPHULL, the current change file, and the file VPROC (which contains all the PROCs produced from files CHAFWL and CHVECT). One change that may be required for a new EPHULL version is the sequence number where these PROCs are to be added.

The previously neglected, single-material (airblast) routines in EPHULL have been rewritten in vector form, to allow organizations to again use a single HULL code to solve virtually all their continuum-mechanics problems.

---

9. Durrett, R. E., et al., **The HULL Hydrodynamics Computer Code**, AFWL-TR-76-183, Air Force Weapons Laboratory, Kirtland AFB, NM, September 1976.

Use of this code should relieve user organizations from the requirement to maintain several specialized versions of the HULL code. This composite code is intended to be maintained under the SAIL executive system (rather than the UPDATE-POST system to which some versions have migrated).

b. Change files--The change file CHVECT consists of over 14,000 lines. It modifies program PLANK to recognize the keyword VECTOR. When running a HULL job, PLANK notifies program SAIL that vector coding is desired by setting the VECTOR option to one in file INPUT2. The vector option has not been added to the Z-BLOCK, and therefore must be specified in HULL input each time a problem is restarted. If the vector option is not specified, the default scalar code is produced by SAIL.

The optimization techniques in CHVECT use the well-publicized idea of innermost do-loop vectorization. On the CRAY computer, only the innermost do-loop will be vectorized, but even this loop will not be vectorized if there is a vector inhibitor present. Vector inhibitors include calls to externals, if-checks inside the loop, and input/output operations. During code checkout, another vector inhibitor was discovered: any if-check branching to the end of an outer do-loop inhibits an inner do-loop using the same statement number as its end.

Many loops in HULL are dual loops: one index is run over the number of materials in the problem, and the other index is run over the number of cells in a row. HULL can presently handle a maximum of 10 materials, but most problems have fewer than 5. The number of cells in a row is completely arbitrary, but most HULL problems have more than 10 cells per row. In the scalar versions of HULL, the outer loops are usually over the cells in a row, and the inner loops are over the materials. Since the largest gain can be realized by having the inner loops operate on the longest possible vectors, the loops in Composite HULL have been inverted. The outer loops in the new code are over the materials, and the inner loops are over the cells in a row. This scheme often results in several additional loops being required, but usually no additional calculations are involved. If a quantity is needed in more than one loop, a vector temporary is employed to store the quantity for subsequent loops.

Most of CHVECT is concerned with modifications to program HULL. Vectorized versions of subroutines EOSSET, BHYDRO, HYDRO, TRHYDRO, THYDRO, BFLUX, FLUX, TRFLUX, BFLUX2, FLUX2, TFLUX2, STATE, and most EOS routines have been provided. Changes were also required in subroutine OUTPUT to make it compatible with the revised EOS common blocks.

In the modified subroutines, cells in a row are scanned to determine a subset of adjacent cells that are "the same" as the current leftmost cell. In some routines "the same" may mean "all fluid cells" or "all island cells" (HYDRO routines): in others "the same" may mean "all containing the same material mix" (FLUX routines, STATE, etc). The actual selection of a subgroup is accomplished by using a new array called ICD, which is dimensioned to IMAX*NROWIC; therefore, an element of this array is available for each cell in core. Subroutine EOSSET is responsible for setting the cell descriptor (ICD) word for each cell at the start of each cycle. If a cell is an island cell (a rigid nonfluid cell), ICD for that cell is set to -1. If a cell contains only one material, ICD is set to 2**IM (where IM is the material number). If the cell is a multimaterial cell, ICD is set to the sum of 2**IM (where IM is the material number for each material in the cell) plus 99000. A new integer function called ICOUNT counts the number of elements in the ICD array that are "the same." ICOUNT requires three arguments: (1) the number of elements to scan; (2) the first element of the array; and (3) the type of scan. There are three types of scans: (1) elements greater than or less than zero (fluid or island cells); (2) positive elements greater than or less than 99000 (mixed- or single-material cells); and (3) exact agreement (same single material or same mixture, or islands).

Once a group has been determined, the cells are processed in vector loops. Since they are all "the same," no decisions have to be made in the loops; therefore, the loops can be vectorized. A further restriction is that groups of adjacent cells cannot be longer than 64. Most of the temporary arrays (vector temporaries) have been dimensioned to 65 words, to minimize the additional memory required. The size is set to 65, to facilitate left-right equivalenced arrays each 64 words long, but offset by one word; for instance: UL(I+1) (horizontal velocity at the left boundary of cell I+1) is the same memory location as UR(I) (the horizontal velocity at the right boundary of cell I). The size of these vector temporaries can be established by setting

the option SIZE in the SAIL input file (Appendix A). For very long rows, some speedup can be realized by setting SIZE equal to IMAX.

The variable names used in the scalar version have been retained as often as possible, but some of these variables (undimensioned variables) have been made into vectors with the same name [example: UL into UL(I), etc.]. The order of calculation has also remained virtually unchanged.

The change file CHAFWL contains coding changes required for AFWL CRAY compatibility. CHAFWL includes the modified BOW program, and changes to all the permanent file manipulation subroutines. The change files CHVECT, CHAFWL, AFWLCH, and VPROC are located on CFS under the /NTEPHULL/SOURCELIB subdirectory (Fig. 1).

c. Latest Update--As of the last edit of this report, Composite HULL may be accessed in Version 120 by using the change files: CH120, CHNEW, and VPROC. These files can be found under both the /NTEPHULL/SOURCELIB and /NTEPHULL/SYSGEN subdirectories.

IV. SAIL CODE

1. BACKGROUND

During initial development of the HULL code in 1971, Daniel Matuska and Richard Durrett (then Captains) created the original SAIL code based on concepts borrowed from Dr. Reginald W. Clemens. These concepts were expanded to include an enlarged syntax, and were combined with an update system in December 1973 by Lt. Lewis Gaby. Since then, SAIL has continued to evolve, and is now essentially a commercial product maintained by OTI, the present employer of Mr. Daniel Matuska (Ref. 10).

The most recent version of SAIL maintained by AFWL was acquired from AFATL in 1981, and modified for CTSS by (then Captain) Raymond L. Bell and Ms. Cydney Westmoreland. The source code (all Fortran) is maintained on CFS, under the /NTEPHULL/SOURCELIB and /NTEPHULL/BACKUP subdirectories as file MSAIL.

2. GENERAL DESCRIPTION

SAIL can be used to maintain virtually any computer file. The procedures to convert a file to a SAIL Library File (SLF) are given in Appendix A. Each program on an SLF is essentially equivalent to a "deck" under UPDATE. Instead of a deck name followed by sequential numbers (as in UPDATE), SAIL uses only sequential numbers for each line on the SLF. To differentiate between programs on an SLF, the sequential numbers for each program start on a multiple of 10,000.

SAIL directives (*KEEPTO, *PRUC, etc.) embedded in the SLF allow users to selectively keep or reject blocks of coding, based on the values of options specified in the SAIL input file (and the file INPUT2, if running with program PLANK). SAIL will make temporary changes to the SLF if changes are in the SAIL input file, or if the input file contains a directive to read a particular change file (*READ FILENAME). Changes can be in several change files; each one must be local, and have a *READ NAME in the SAIL input file. The changes need not be in sequence. SAIL will issue warning messages if any of the requested changes conflict with each other.

---

10. **SAIL Users Guide for Running the HULL and EPIC3 Codes**, Orlando Technology, Inc., 60 Second St., Bldg. 5, Shalimar, FL.

In NORMAL mode, SAIL will process the SLF using the default option values on the SLF (and additional option values specified by the user), to produce one or more source code files (named SAIL, SAIL1, SAIL2, etc.), and an output file describing the SLF that was processed along with any SAIL diagnostic messages.

3. USING SAIL ON THE AFWL CRAY

All files (except for change files) required for running the HULL system at AFWL are in a library file known as HLIBA. This library file is on CFS, under subdirectories /NTEPHULL/SOURCELIB and /NTEPHULL/BACKUP. HLIBA contains: (1) executable programs BOW, PLANK, and XSAIL (executable SAIL), (2) files OLD (the SLF for EPHULL) and MATLIB (the material property library), and (3) another library file (HULLIB), which contains relocatable utility routines needed to load newly compiled HULL system programs.

To use SAIL on the AFWL CRAY, a user must first get HLIBA from CFS, then enter the following CTSS commands:

LIB HLIBA <esc> X ALL. <esc> END <carriage return>

where <esc> = escape key. The user's file space will now contain all the files listed above.

The user can produce an input file for SAIL using any editor available on CTSS. The format for SAIL input files is given in Appendix A. In general, the file is free-format, but the word SAIL must be the first word on the file. The file can be given any name. If SAIL is executed without specifying particular names for the input and output files, the input file must be called INPUT, and the output file produced will be called OUTPUT. To specify particular names for the input and output files, type: XSAIL I=INNAME O=OUTNAME, where INNAME is the input file name and OUTNAME is the desired output file name. More detailed procedures are given in Appendix A. Examples in the next section show how SAIL is used in the NORMAL mode.

V. EXAMPLE PROBLEMS

1. COSMOS FILES

The COSMOS files shown in Tables 1, 2, 3, and 4 are set up for jobs being submitted from a Remote Job Entry Terminal (RJET). The first two lines on each file are Control Data Corporation (CDC) Job Control Language (JCL) giving the CDC job name, time limit, priority, and the instruction to stage the job to the CRAY (STCRA). The second line is the CDC account card. The third line on each file is the CRAY job card. This card contains the CRAY job name, user name, password, CRAY account number, job time limit, class, and priority.

The next line is the COSMOS command

```
*INTERRUPT ON SOFTWAREERROR TO EXIT
```

which causes the COSMOS job stream to branch to the COSMOS line

```
*EXIT:
```

if an error occurs while executing any line before the *EXIT: label.

The next COSMOS command line

```
*MASS GET DIR=/NTEPHULL/SOURCELIB HLIBA CH120 VPROC CHNEW
```

causes the files HLIBA, CH120, and VPROC to be copied from CFS into the local disk space reserved for this job.

The next COSMOS command line

```
*LIB HLIBA
```

executes the CTSS utility LIB. The next two lines are commands to LIB; therefore, they do not have an asterisk (*) in the first column

```
X ALL.
```

TABLE 1. KEEL COSMOS JOB FILE

```
KEEL,TXX,PXX,STCRA.
ACCOUNT(NAME,OOOOXXXX-NED,NTEDA,PHONE)
*/JOB US=2318,PW=XXXXXX,CC=0000####,TL=30.0,PR=1,CL=C
*INTERRUPT ON SOFTWAREERROR TO EXIT
*MASS GET DIR=/NTEPHULL/SOURCELIB HLIBA CH120 VPROC CHNEW
*LIB HLIBA
 X ALL.
 END
*FILE NAME=INPUT,END=EOR
 SAIL LINENO
*READ CH120
*READ VPROC
*READ CHNEW
EOR
*FILE NAME=KEELIN,END=EOR (KEEL input file--See example problems)
EOR
*BOW I=KEELIN O=BOWOUT
*PLANK I=KEELIN O=PLNKOUT
*XSAIL I=INPUT O=SAILOUT
*INTERRUPT ON SOFTWAREERROR TO SUNK
*CFT I=SAIL,B=BKEEL,L=KEELLST,ON=DINXZA
*LDR BIN=BKEEL,LIB=(HULLIB,CFTMATH),MO=FULL,ML=KEELMAP,X=XKEEL
*XKEEL I=KEELIN O=KEELOUT
*GOTO EXIT
*SUNK:
*CONCAT OUT1 KEELLST KEELMAP
*DISPOSE DN=OUT1,FID=SINKLST
*EXIT:
*CONCAT OUT2/CC BOWOUT/CC PLNKOUT/CC SAILOUT/CC KEELOUT/CC
*SELECT PRINTLOG=KEELOG
*CONCAT KEELOT/CC OUT2/CC KEELOG KEELOUT/CC
*DISPOSE DN=KEELOT,FID=KEELOUT,SF=CC
```

TABLE 2. HULL COSMOS JOB FILE

```
HULL,Pxx,Txx,STCRA.
ACCOUNT(NAME,OOOOXXXX-NED,NTEDA,PHONE)
*/JOB US=2318,PW=XXXXXX,CC=0000####,TL=30.0,PR=1,CL=C
*INTERRUPT ON SOFTWAREERROR TO EXIT
*MASS GET DIR=/NTEPHULL/SOURCELIB HLIBA CH120 VPROC CHNEW
*LIB HLIBA
 X ALL.
 END
*FILE NAME=INPUT,END=EOR
 SAIL LINENO
*READ CH120
*READ VPROC
*READ CHNEW
EOR
*FILE NAME=HULLIN,END=EOR (HULL input file--See example problems)
EOR
*BOW I=HULLIN O=BOWOUT
*PLANK I=HULLIN O=PLNKOUT
*XSAIL I=INPUT O=SAILOUT
*INTERRUPT ON SOFTWAREERROR TO SUNK
*CFT I=SAIL,B=BHULL,L=HULLLST,ON=DINXZA
*LDR BIN=BHULL,LIB=(HULLIB,CFTMATH),MO=FULL,ML=HULLMAP,X=XHULL
*XHULL I=HULLIN O=HULLOUT
*GOTO EXIT
*SUNK:
*CONCAT OUT1 HULLLST HULLMAP
*DISPOSE DN=OUT1,FID=SINKLST
*EXIT:
*CONCAT OUT2/CC BOWOUT/CC PLNKOUT/CC SAILOUT/CC
*SELECT PRINTLOG=HULLLOG
*CONCAT HULLOT/CC OUT2/CC HULLLOG
*DISPOSE DN=HULLOT,FID=HULLOUT,SF=CC
```

TABLE 3. PULL COSMOS JOB FILE

```
PULL,Pxx,Txx,STCRA.
ACCOUNT(NAME,OOOOXXXX-NED,NTEDA,PHONE)
*/JOB US=2318,PW=XXXXXX,CC=0000####,TL=30.0,PR=1,CL=C
*INTERRUPT ON SOFTWAREERROR TO EXIT
*MASS GET DIR=/NTEPHULL/SOURCELIB HLIBA CH120 VPROC CHNEW
*LIB HLIBA
 X ALL.
 END
*FILE NAME=INPUT,END=EOR
 SAIL LINENO
*READ CH120
*READ VPROC
*READ CHNEW
EOR
*FILE NAME=PULLIN,END=EOR (PULL input file--See example problems)
EOR
*BOW I=PULLIN O=BOWOUT
*PLANK I=PULLIN O=PLNKOUT
*XSAIL I=INPUT O=SAILOUT
*INTERRUPT ON SOFTWAREERROR TO SUNK
*CFT I=SAIL1,B=BPULL,L=PULLLST,ON=DINXZA
*LDR BIN=BPULL,LIB=(HULLIB,CFTMATH,METALIB),
    MO=FULL,ML=PULLMAP,X=XPULL
*XPULL I=PULLIN O=PULLOUT
*MASS STORE PLTFILE:/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx/PULLPLOTS
*GOTO EXIT
*SUNK:
*CONCAT OUT1 PULLLST PULLMAP
*DISPOSE DN=OUT1,FID=SINKLST
*EXIT:
*CONCAT OUT2/CC BOWOUT/CC PLNKOUT/CC SAILOUT/CC
*SELECT PRINTLOG=PULLLOG
*CONCAT PULLOT/CC OUT2/CC PULLOG PULLOUT/CC
*DISPOSE DN=PULLOT,SF=CC,FID=PULLOUT
```

TABLE 4. STATION COSMOS JOB FILE

```
STAT,Pxx,Txx,STCRA.
ACCOUNT(NAME,OOOOXXXX-NED,NTEDA,PHONE)
*/JOB US=2318,PW=XXXXXX,CC=0000####,TL=30.0,PR=1,CL=C
*INTERRUPT ON SOFTWAREERROR TO EXIT
*MASS GET DIR=/NTEPHULL/SOURCELIB HLIBA CH120 VPROC CHNEW
*LIB HLIBA
 X ALL.
 END
*FILE NAME=INPUT,END=EOR
 SAIL LINENO
*READ CH120
*READ VPROC
*READ CHNEW
EOR
*FILE NAME=STATIN,END=EOR (STATION input file--See example problems)
EOR
*BOW I=STATIN O=BOWOUT
*PLANK I=STATIN O=PLNKOUT
*XSAIL I=INPUT O=SAILOUT
*INTERRUPT ON SOFTWAREERROR TO SUNK
*CFT I=SAIL,B=BSTAT,L=STATLST,ON=DINXZA
*LDR BIN=BSTAT,LIB=(HULLIB,CFTMATH,METALIB),
     MO=FULL,ML=STATMAP,X=XSTAT
*XSTAT I=STATIN O=STATOUT
*MASS STORE PLTFILE:/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx/STAPLOTS
*GOTO EXIT
*SUNK:
*CONCAT OUT1 STATLST STATMAP
*DISPOSE DN=OUT1,FID=SINKLST
*EXIT:
*CONCAT OUT2/CC BOWOUT/CC PLNKOUT/CC SAILOUT/CC
*CONCAT STATOT/CC OUT/CC STATLOG STATOUT/CC
*DISPOSE DN=STATOT,SF=CC,FID=STATOUT
```

extracts all files from HLIBA, and

```
END
```

terminates LIB.

The next COSMOS command line

```
*FILE NAME=INPUT,END=EOR
```

causes the creation of a local file named INPUT consisting of the lines following this line down to, but not including, the EOR line. The lines in file INPUT

```
SAIL LINENO
*READ CH120
*READ VPROC
*READ CHNEW
```

were explained above with the exception of LINENO. The LINENO parameter causes SAIL to append the sequence numbers from the SLF onto the lines of the source code files that SAIL produces.

The next COSMOS command line

```
*FILE NAME=NNNNIN,END=EOR
```

(where NNNN=KEEL, HULL, PULL, or STAT, depending on which COSMOS file is being run) causes creation of file NNNNIN, which consists of all lines following this line down to, but not including, the EOR line.

The next COSMOS command line

```
*BOW I=NNNNIN O=BOWOUT
```

(where NNNNIN is defined above) executes program BOW using NNNNIN as input, and creating BOWOUT as output. The actions accomplished by program BOW for each COSMOS file are different, and are explained below.

The next COSMOS command line

```
*PLANK I=NNNNIN O=PLNKOUT
```

(where NNNNIN is defined above) executes program PLANK using NNNNIN as input and creating PLNKOUT as output. PLANK determines the program(s) and options desired by the user, and produces file INPUT2 for program SAIL. Specific actions accomplished by program PLANK for each COSMOS file are explained below.

The next COSMOS command line

```
*XSAIL I=INPUT O=SAILOUT
```

executes program SAIL. The executable code is named XSAIL to differentiate it from the source code files named SAIL, SAIL1, SAIL2, etc. SAIL reads INPUT2 which indicates, through the option/value pairs, which program is to be created, what the array sizes are to be, what program options are to be included, and what materials are involved. Program SAIL also reads file INPUT, which indicates that line numbers are to be appended to each line of the source code as it is being created, and that changes are contained in files CH120, VPROC, and CHNEW.

SAIL reads the change files and places the sorted changes in local file CHANGE. SAIL then begins processing OLD (the SLF) along with the changes and the defined options to produce file SAIL, which is a customized source code for the appropriate program (KEEL, HULL, PULL, or STATION). Program SAIL then writes a summary on file SAILOUT along with any error or warning messages. If any fatal SAIL errors have occurred, SAIL calls abort; otherwise, it terminates normally.

If an error had occurred in any of the above COSMOS command lines, the job stream would have branched directly to the *EXIT: line. The next COSMOS command line

```
*INTERRUPT ON SOFTWAREERROR TO SUNK
```

changes the destination label to which COSMOS will branch if an error is detected from *EXIT: to *SUNK:.

The next COSMOS command line

```
*CFT I=SAIL,B=BNNNN,L=NNNNLST,ON=DINXZA
```

(where NNNN=KEEL, HULL, PULL, or STAT depending on which COSMOS file is being run) executes the CRAY Fortran Compiler (CFT). CFT uses file SAIL as the source code. The resulting binary code is placed in file BNNNN (NNNN defined above), and the program listing is placed in file NNNNLST. The compiler option list (ON=DINXZA) generates a comprehensive cross-reference map and list of variables after each subroutine listing in NNNNLST.

The next COSMOS command line

```
*LDR BIN=BNNNN,LIB=(HULLIB,CFTMATH),MO=FULL,ML=NNNNMAP,X=XNNNN
```

(where NNNN is defined above) executes the CRAY Loader (LDR). LDR uses the binary file BNNNN (produced by CFT) and the utility and math routines contained in HULLIB and the CTSS library CFTMATH, to produce the executable program XNNNN. The full loadmap is placed in file NNNNMAP (where NNNN is defined above). The system library METALIB is also required for programs producing graphics output (PULL and STATION).

The next COSMOS command line

```
*XNNNN I=NNNNIN O=NNNNOUT
```

(where NNNN is defined above) executes program NNNN. The actions accomplished by each program are explained below. If an error is detected during execution of CFT, LDR, or XNNNN, the job stream branches to label *SUNK:. If XNNNN terminates normally, the next COSMOS command line

```
*GOTO EXIT
```

executes, and the job stream branches to the label *EXIT:.

The next COSMOS command line is the label

```
*SUNK:
```

which is the destination if an error is detected during execution of CFT, LDR, or XNNNN. The COSMOS commands following this line will be executed only if the job stream has been branched here. The next two COSMOS command lines

```
*CONCAT OUT1 NNNNLST NNNNMAP
```

(where NNNN is defined above) and

```
*DISPOSE DN=OUT1,FID=SINKLST
```

provide the user with the listings and loadmaps that may help determine what went wrong.

The next COSMOS command line

```
*EXIT:
```

is the destination label if an error was detected in the lines preceding the second *INTERRUPT command line, or if the job stream was branched due to the *GOTO EXIT command line.

The remaining COSMOS command lines

```
*CONCAT OUT2/CC BOWOUT/CC PLNKOUT/CC SAILOUT/CC NNNNOUT/CC
*SELECT PRINTLOG=NNNNLOG
*CONCAT NNNNOT/CC OUT2/CC NNNNLOG
*DISPOSE DN=NNNNOT,FID=NNNNOUT,SF=CC
```

(where NNNN is KEEL, HULL, PULL, or STAT) provide the printed output from the executed programs and a record of the COSMOS commands that were executed. These output files can then be physically picked up at the AFWL central computing facility.

The operation of programs BOW, PLANK, KEEL, HULL, PULL, and STATION (Tables 1-4) is described in detail below.

a. KEEL job file--Program BOW reads input file KEELIN, to determine the problem number and whether the problem is being rerun. BOW creates and attempts to SAVE file BUSYBIT under subdirectory /NTEPHULL/EPHULLPROBLIB as described above in Section II. When BUSYBIT is successfully saved, BOW gets file BOWDATABASE from this same subdirectory. BOWDATABASE is a sequential listing of problem numbers that presently exist on CFS. BOW compares this problem number with those listed on BOWDATABASE.

If this problem number is not found on BOWDATABASE, it is inserted in sequence and a new subdirectory node is added on CFS under subdirectory /NTEPHULL/RESTARTFILESC with the name PxxxxxPxxxx (where xxxxx.xxxx is this problem number). BOW then creates file BOWBUSY and saves it under this new subdirectory. The revised BOWDATABASE file is stored under subdirectory /NTEPHULL/EPHULLPROBLIB.

If this problem number is found on BOWDATABASE and the keyword RERUN is not specified on file KEELIN, BOW writes a message on BOWOUT and calls abort.

If this problem number is found on BOWDATABASE and the keyword RERUN is specified on KEELIN, BOW creates file BOWBUSY and attempts to save it under subdirectory /NTEPHULL/RESTARTFILESC/PxxxxxPxxxx (where xxxxx.xxxx is this problem number). If an error occurs, BOW writes a message on BOWOUT and calls abort. If the save is successful, BOW gets the file BOWDATA from this problem subdirectory. BOW reads BOWDATA to determine the restart and station file names stored under this subdirectory. BOW forms the path names for each of these files and deletes them from CFS.

Finally BOW creates a file named KEELMSG which contains the following:

```
BOWDATA FOR PROB xxxxx.xxxx
NSTA          MM/DD/YY  HH:MM:SS
T4C00000        0.0         0
LAST
```

where xxxxx.xxxx is this problem number, and where NSTA=NULL if this problem does not have stations, or T9C00000 if it does. The current date and time are indicated by MM/DD/YY (month/day/year) and HH:MM:SS (hour:minute:second). This file is used by program KEEL as the initial BOWDATA file.

BOW then deletes BUSYBIT from CFS. Note that BOW does not delete BOWBUSY. The presence of BOWBUSY assures that no other job will be able to modify files under this problem subdirectory until KEEL finishes successfully. The last thing program KEEL does, is delete BOWBUSY. BOW writes a summary message on file BOWOUT and terminates.

Program PLANK reads file KEELIN. The option/value pairs in KEELIN and the program name (KEEL) are written on file INPUT2. PLANK writes a summary message on file PLNKOUT and terminates.

Program KEEL reads file KEELIN, to set the problem number, to determine problem mesh coordinates, to place the appropriate materials into the mesh, to determine initial conditions (velocities, energies, etc.), and to establish the initial locations of any included stations. KEEL then creates the initial restart file (TAPE4) and stores it under the problem subdirectory as file T4C00000. If stations are included, the initial station file (TAPE9) is created and stored under the problem subdirectory as file T9C00000. File KEELMSG is then stored under the problem subdirectory as file BOWDATA, and file BOWBUSY is deleted. Program KEEL writes out the options set for this problem (the Z-BLOCK), the mesh coordinates, material properties for the materials used, descriptions of where various materials were placed in the mesh, initial locations of any stations generated, and a material map of the mesh. If an error is detected during execution, program KEEL calls abort without deleting BOWBUSY, and the COSMOS job stream branches to *SUNK:.

b. HULL job file--Program BOW reads input file HULLIN, to determine the problem number and the restart conditions (the default is a restart, from the last restart file available). BOW gets file BOWDATABASE from subdirectory /NTEPHULL/EPHULLPROBLIB without unique access (BOW does not create and save file BUSYBIT), since file BOWDATABASE will not be modified.

BOW compares this problem number with those listed on file BOWDATABASE. If the problem number is not found on BOWDATABASE, BOW writes a message on BOWOUT and calls abort. If the problem number is found on BOWDATABASE, BOW creates and attempts to save file BOWBUSY under subdirectory /NTEPHULL/RESTARTFILESC/PxxxxxPxxxx (where xxxxx.xxxx is the problem number). If an error occurs on the save attempt, BOW writes a message on file BOWOUT and calls abort. If the save is successful, BOW gets the BOWDATA file from the problem subdirectory.

As shown in Table 5, BOWDATA contains a list of restart file names with the problem time and cycle associated with each.

TABLE 5. BOWDATA FILE STRUCTURE

BOWDATA FOR PROB xxxxx.xxxx		
NSTA	MM/DD/YY	HH:MM:SS
T4C00000	0.0	0
T4C00100	time1	cycle1
T4C00200	time2	cycle2
"	"	"
"	"	"
"	"	"
T4C00N00	timeN	cycleN
LAST		

If a restart time or cycle is specified on file HULLIN, BOW compares the times (or cycles) on file BOWDATA to determine the proper restart file name. All subsequent restart files are deleted from CFS, and the BOWDATA file is revised and stored on CFS. If no restart time or cycle is specified, BOW assumes the problem is to be restarted from the latest restart file.

BOW creates file HULLMSG, which consists of the first two, and last two, lines from file BOWDATA. HULLMSG provides the restart file name to program HULL, so that subsequent restart file names are properly sequenced. BOW gets the appropriate restart file from CFS, with the local file name TAPE4. If the problem contains stations or particles, BOW also gets the station file (T9C00000), with the local file name TAPE99.

As with KEEL, BOW does not delete BOWBUSY. The presence of BOWBUSY assures that no other job can modify files for this problem until program HULL finishes successfully and then deletes file BOWBUSY from CFS. BOW writes a summary message on file BOWOUT, and terminates.

Program PLANK reads the options and values from the Z-BLOCK record on TAPE4 (the restart file) and any option/values on file HULLIN. PLANK writes these option name/value pairs and the program name (HULL), on file INPUT2 for program SAIL. PLANK writes a summary message on file PLNKOUT and terminates.

Program HULL reads file HULLIN to determine if any parameters are to be changed. HULL reads all of TAPE4 to set the coordinate arrays, the Z-BLOCK variables, the material properties, and the mesh values for the problem restart time. If TAPE99 is present, HULL creates a local file TAPE99, and copies TAPE9 to it. HULL then reads through TAPE9 to the appropriate restart time.

HULL then proceeds to advance the mesh through time. Periodically, HULL produces new restart files (based on the values of "dump" parameters). These new restart files are written on a newly created local file TAPE4. HULL stores this new TAPE4 under the problem subdirectory with the name T4Cxxx00 (where xxx is the next consecutive number between 001 and 999). If the problem has a station file (TAPE9), HULL copies the current TAPE9 to a newly created file (TAPE99), and stores this file under the problem subdirectory with the name T9C00000. HULL then updates and stores the BOWDATA file. Upon reaching the desired problem stop time (or detecting some other stop condition), HULL produces and stores a restart file (and station file) as described above, stores the revised BOWDATA file, deletes BOWBUSY, and then terminates. Each time a restart file is created, HULL produces a significant amount of printed output, which includes current values of the Z-BLOCK variables, job statistics, and material and energy maps. If an error is detected during execution, HULL writes a message on file HULLOUT describing the nature of the error, and then calls abort. In this case, HULL does not delete BOWBUSY, and the COSMOS job stream branches to *SUNK:.

c. PULL job file--Program BOW reads input file PULLIN, to determine the problem number and plot times. BOW gets BOWDATABASE from subdirectory /NTEPHULL/EPHULLPROBLIB without unique access, since the file will not be modified. BOW compares the problem number with those listed on file BOWDATABASE. If the problem number is not found on BOWDATABASE, BOW writes a message on file BOWOUT, and calls abort. If this problem number is found on BOWDATABASE, BOW attempts to get file BOWDATA from the problem subdirectory. BOW does not attempt to save BOWBUSY under the problem subdirectory, since PULL does not modify any files under this subdirectory. As shown (Table 5), BOWDATA contains a list of restart file names, with the problem time and cycle associated with each. Based on the plot times requested in file PULLIN, BOW writes the required restart file names on file PULLMSG. BOW then gets the first restart file listed on PULLMSG, writes a summary message on file BOWOUT, and terminates.

Program PLANK reads the options and values from the Z-BLOCK record on TAPE4, and any option/value pairs on file PULLIN. PLANK writes these option/value pairs and the program name (PULL) on file INPUT2 for program SAIL, writes a summary message on file PLNKOUT, and then terminates.

Program XPULL reads file PULLIN, to determine what plots are required at each plot time. The required restart file names for each time are read from file PULLMSG. XPULL assumes that the existing TAPE4 is the first restart file listed on PULLMSG. After completing all plots requested for a plot time, XPULL destroys TAPE4, gets the next restart file listed on PULLMSG (if any) as TAPE4, and produces plots requested for this plot time. After completing all requested plots, XPULL writes a summary of the plots produced, and terminates. The file containing the plots (PLTFILE) is then stored, under the problem subdirectory with the name PULLPLOTS. For color plotting jobs, the file containing the plots is called TAPE89.

d. STATION job file--Program BOW reads input file STATIN, to determine the problem number. BOW gets BOWDATABASE from directory /NTEPHULL/EPHULLPROBLIB without unique access, since the file will not be modified. BOW compares the problem number with those listed on BOWDATABASE. If the problem number is not found on BOWDATABASE, BOW writes a message on BOWOUT, and calls abort. If the problem number is found on BOWDATABASE, BOW attempts to get file

BOWDATA from the problem subdirectory. BOW does not attempt to save BOWBUSY under the problem subdirectory, since STATION does not modify any files under this subdirectory. As Table 5 shows, BOWDATA contains the station file name if the problem has stations. If the station file name found on BOWDATA is NULL, BOW writes a message on file BOWOUT, and calls abort. If the station file name is T9C00000, BOW gets the station file as TAPE4, writes a summary message on file BOWOUT, and then terminates.

Program PLANK reads the options and values from the Z-BLOCK record on TAPE4 (the station file), and from any option/value pairs on file STATIN. PLANK writes these option name/value pairs and the program name (STATION) on file INPUT2 for program SAIL, writes a summary message on file PLNKOUT, and then terminates.

Program XSTAT reads file STATIN to determine what plots are required for particular stations. XSTAT produces the requested plots for each of any requested stations, writes a summary on file STATOUT, and then terminates. File PLTFILE is then stored under the problem subdirectory as STAPLOTS.

2. ADDITIONAL EXAMPLES

Due to a scarcity of computer funds, the planned example problems for nuclear airblast, high explosive burn, and penetration (with STRESS 1) have been delayed. AFWL plans to run three example plans in the near future, and to publish the results in an AFWL technical report. The following four tables (Tables 6-9) contain typical input files for KEEL, HULL, PULL, and STATION jobs.

TABLE 6. KEEL INPUT FILE

```
KEEL PROB 1.1
STRESS=0 IMAX=100 JMAX=100 NM=3 NSTN=10 NOP=1
AIR=1 TNTBRN=2 TNT=3 BURN=1 DIMEN=2
HEADER
TEST PROBLEM
MESH
XO-=0.0 XMAX=100.0
YO=0.0 YMAX=100.0
GENERATE
PACKAGE AIR RECTANGLE
DELETE CIRCLE RAD=25.0
PACKAGE TNT CIRCLE RAD=25.0
DELETE CIRCLE RAD=3.0
PACKAGE TNTBRN CIRCLE RAD=3.0
STATIONS
XS=1.0 2.0 20.0 30.0
YS=1.0 2.0 20.0 30.0
```

TABLE 7. HULL INPUT FILE

```
HULL PROB 1.1
INPUT VECTOR
TIMES=3 DMDINT=5.0e-3
PTSTOP=3.0e-3
```

TABLE 8. PULL INPUT FILE

```
PULL PROB 1.1
VELVECT PHHST
PCONT DCONT
```

TABLE 9. STATION INPUT FILE

```
PULL STATION PROB 1.1
STATIONS ALL ALL
END
```

VI. OTHER CODE MODIFICATIONS

1. CIST EQUATION OF STATE

The CIST EOS, in the CRALE Code (Ref. 1), was rewritten to match the structure and variable names used in HULL. Essentially, this EOS relates pressure (stress) and volumetric strain, where the pressure is calculated as a linear function of the volumetric strain. The particular linear function changes at specific values of volumetric strain (up to seven different linear segments), then transitions to a quadratic function at some limiting value of volumetric strain (Fig. 2).

When the material is being loaded (density is increasing as the calculation advances), the pressure is calculated as

$$P = PL(i)+BL(i)*(U-UL(i)) \tag{1}$$

where

P = Pressure corresponding to U
BL(i) = Bulk modulus for Segment i (segment containing U)
UL(i) = Volumetric strain at left end of Segment i
PL(i) = Pressure on loading curve corresponding to UL(i)
U = Current value of volumetric strain for material in the cell of interest
U = Rho/Rho0 - 1.0
Rho0 = Ambient density
Rho = Current density

as long as U is between UL(i) and UL(i+1). Once U exceeds the linear limit [at most UL(7)], the pressure is calculated as

$$P = Ptran+Btran*(U-Utran)+Bsq*(U-Utran)**2 \tag{2}$$

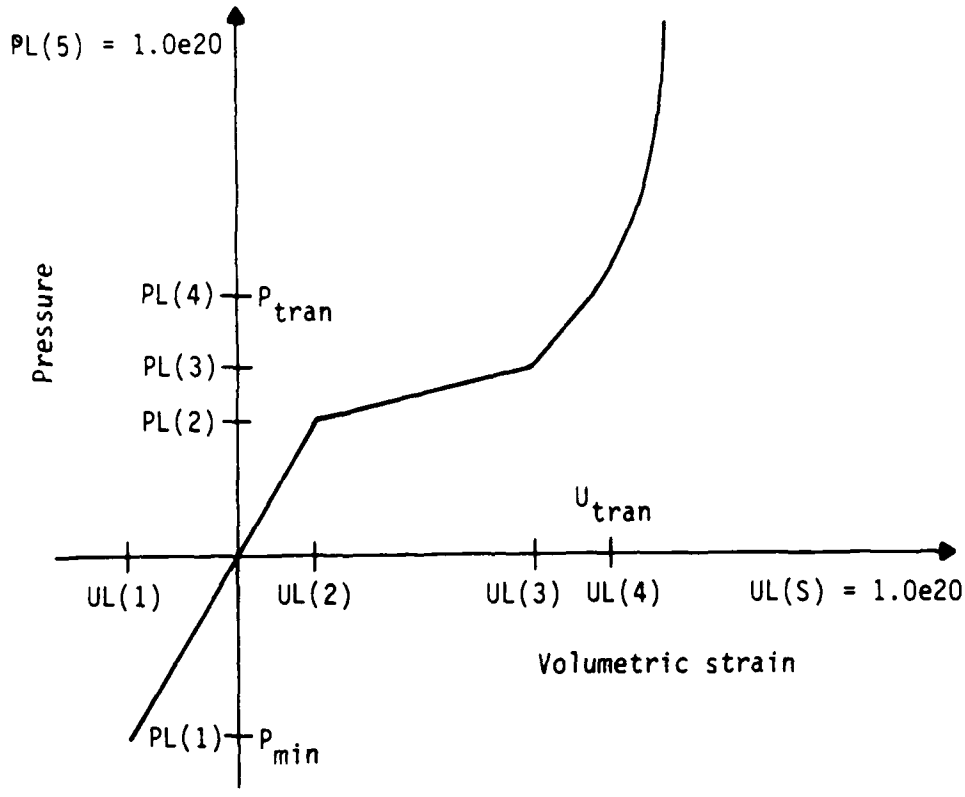


Figure 2. CIST EOS loading curve.

where

Utran = Volumetric strain where quadratic curve begins
Ptran = Pressure corresponding to Utran
Btran = Linear coefficient for quadratic curve
Bsq = Quadratic coefficient

The unloading curve (Fig. 3) for this cell is assumed to attach to its loading curve (Fig. 2) at the point of maximum loading achieved. When the material is being unloaded (ρ decreasing as the calculation advances), the pressure is based on the appropriate unloading bulk modulus.

The index for the unloading segment is determined by calculating the limiting volumetric strains, based on the unloading pressures and the maximum volumetric strain for this cell (Umax, the point where the unloading curve attaches to the loading curve). The pressure corresponding to the current volumetric strain is then

$$P = PU(i)+BU(i)*(U-UU(i)) \tag{3}$$

where

P = New pressure
UU(i) = Left end of Segment i (segment containing U)
PU(i) = Pressure corresponding to UU(i)
BU(i) = Bulk modulus on unloading segment i
U = Current value of volumetric strain

The unloading curve is also used for reloading the material, as long as U is less than Umax. Once U exceeds Umax, the loading curve is used again. If U then decreases, a new unloading curve is established.

The sound speed is calculated from the equation for bulk modulus

$$B = (1/3)(Rho\ C**2)(1+NU)/(1-NU) \tag{4}$$

$$Cnew = (3*B/Rho)(1-NU)/(1+NU) \tag{5}$$

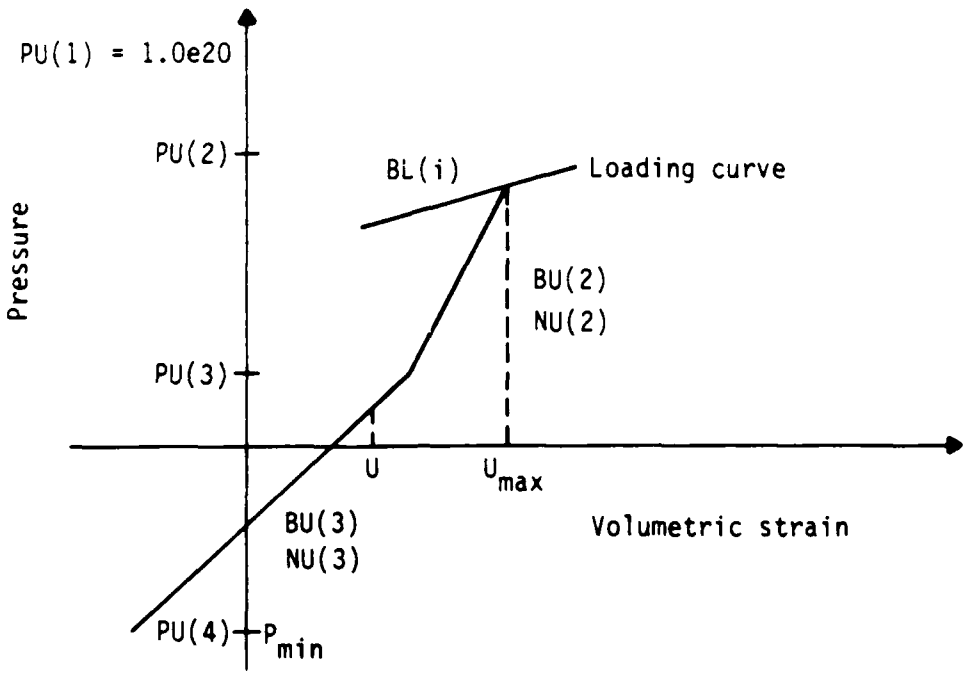


Figure 3. CIST EOS unloading curve.

where

Cnew = C**2
C = Sound speed
B = Bulk modulus for segment containing U
NU = Poisson's ratio
Rho = Material density

For the quadratic portion of the loading curve an effective B is defined

Beff = Btran+2.0*Bsq(U-Utran) (6)

The quantity Rho*C**2 (variable name RHOCSQ) is calculated

RHOCSQ = Rho*Cnew (7)

The sound speed variable (CN) is set equal to the square root of Cnew

CN = SQRT(Cnew) (8)

For mixed cells, the variable DPDTAU is required and is calculated as

DPDTAU = -(Beff*Rho**2)/Rho0 (9)

The stress deviators are calculated using the modulus

G1 = 1.5*(Beff*(1.0-2.0*NU))/(1+NU) (10)

then

DELTXX = 2.0*G1*DT*DEXX (11)

where

DELTXX = Increment in XX stress
DT = Current time step
DEXX = XX Strain rate

The stress deviators are then calculated

$$DTXX = DTXXO+DELTXX \tag{12}$$

where

DTXXO = Previous value of DTXX

Note: Special corrections are applied to account for rotation in two-dimensional cylindrical geometry. See actual code for equations.

Before returning to the calling routine, the stress deviators calculated above are tested against the Von Mises yield criteria. The flow stress (YLDD) is calculated, using the hydrostatic pressure (P calculated above) and the yield curve parameters. The second stress invariant (YTEST) is calculated, using the new stress deviator values calculated above. If the resultant value of YTEST exceeds YLDD, the deviators are modified by the value (YLDD/YTEST)

$$DTXX = DTXX*(YLDD/YTEST) \tag{13}$$

Upon completing these calculations, the program returns to the calling routine.

2. CONTROL FEATURE

The AFWL CRAY and other computers do not report the status of long calculations to the user, between restart files. This problem is especially difficult for the user if the calculation is a batch job, rather than one being run interactively.

To respond to user desires to conserve scarce computer funds, a new capability known as CONTROL was created. This capability is provided by an additional subroutine in HULL, called CNTROL. Subroutine CNTROL looks for a file named HULLCNTRL under the problem subdirectory every ICNTRL-th cycle. If found, CNTROL gets it, and then deletes it from CFS to assure that it will not be there next time. CNTROL reads the file and follows the instructions found. HULLCNTRL can turn the autopriority option on or off, change the priority,

change the time limit, change the dump interval, turn on Switch 1 to terminate the job, etc. CNTROL then writes a file called HULLSTATUS consisting of the current priority, current time limit, status of autopriority, estimated whiz factor, etc., and the current Z-BLOCK. This file is produced every ICNTRL-th cycle, regardless of whether HULLCNTRL was found or not. HULLSTATUS is then stored under the problem subdirectory. Any user can get HULLSTATUS, and determine how the calculation is progressing. This procedure is similar to the MONITOR capability available at AFATL.

The file format and instructions for creating HULLCNTRL are contained in Appendix B.

The appendices at the end of this report are intended to be used as stand-alone references, to answer most user questions. Additional information is available in the reports listed under REFERENCES.

VII. CONCLUSIONS AND RECOMMENDATIONS

This effort has produced an efficient, easy to read, vectorized version of the Elastic/Plastic HULL hydrodynamics code that may be used on any machine. On the AFWL CRAY, the vectorized version used less than 20 percent of the CPU time on a multimaterial problem than did the original scalar version. The Lagrangian options have not been modified. If these options are to be used by AFWL on a frequent basis, a follow-on effort to vectorize the Lagrangian options should be undertaken.

REFERENCES

1. Schuster, S., **Crale Users Manual**, AFWL-TR-82-45, Air Force Weapons Laboratory, Kirtland AFB, NM, September 1982.

2. **CTSS MINI-REFERENCE**, Los Alamos National Laboratory, August 1984.

3. Bell, R. L., **Cray Time Sharing System (CTSS) Version of the HULL File Maintenance Program -- BOW**, AFWL-TR-84-58, Air Force Weapons Laboratory, Kirtland AFB, NM, August 1984.

4. Bell, R. and Westmoreland, C., **Elastic/Plastic HULL (EPHULL) Operation on the CRAY Time Sharing System (CTSS)**, AFWL-TR-83-6, Air Force Weapons Laboratory, Kirtland AFB, NM, April 1983.

5. Durrett, R. E. and Matuska, D. A., **The HULL Code Finite Difference Solution to the Equations of Continuum Mechanics**, AFATL-TR-78-125, Air Force Armament Laboratory, Eglin AFB, FL, November 1978.

6. Matuska, D. A., and Osborn, J., **HULL Documentation**, Orlando Technology, Inc., 60 Second St., Bldg. 5, Shalimar, FL.

7. Vector HULL Report by Hasdal, unpublished, Air Force Weapons Laboratory.

8. Needham, C. E., Havens, M. L., and Knauth, C. S., **Nuclear Blast Standard (1 kt)**, AFWL-TR-73-55, Air Force Weapons Laboratory, Kirtland AFB, NM, April 1975.

9. Durrett, R. E., et al., **The HULL Hydrodynamics Computer Code**, AFWL-TR-76-183, Air Force Weapons Laboratory, Kirtland AFB, NM, September 1976.

10. **SAIL Users Guide for Running the HULL and EPIC3 Codes**, Orlando Technology, Inc., 60 Second St., Bldg. 5, Shalimar, FL.

APPENDIX A

AFWL SAIL CODE SUMMARY

CONTENTS

CONTENTS (Concluded)

PREFACE

The information in this appendix was obtained primarily from a draft copy of the **SAIL User Guide for Running the HULL and EPIC3 Codes,** written by Orlando Technology, Inc. (OTI) (Ref. A1). Information not applicable to the AFWL CRAY (mostly CDC machine-dependent parameters) has been excluded. The SAIL code used at AFWL is not supported by OTI, because it was not purchased from that company. Any deficiencies in the AFWL SAIL code should be addressed to the Atmospheric Phenomenology Section of the Air Force Weapons Laboratory (AFWL/NTEDA) for corrective action.

A1. **Sail Users Guide for Running the HULL and EPIC3 Codes** by Orlando Technology, Inc., 60 Second Street, Bldg. 5, Shalimar, FL 32579 (draft of report for BRL), no date.

A-I. SAIL FILE LIBRARY FORMAT

The SAIL Code may be thought of as an automated editor. It operates on a file which is in the format described below. That file, produced by SAIL in the Generate, Copy, Convert, or Update mode, is referred to here as the SAIL Library File (SLF). The SLF consists of a header, an option directory, and data.

1. HEADER RECORD

The first record on the SLF (the header record) contains the information listed in Table A1. The header record cannot be longer than 250 words. Because of this limitation, an SLF can have a maximum of 50 programs, and no more than 46 default programs.

TABLE A1. SLF HEADER RECORD

Word	Contents	Type
1	Version number	Integer
2	System name	Character
3	Creation date	Character
4	Number of default programs	Integer
5	First default program name	Character
6	Second default program name	Character
"	"	"
"	"	"
"	"	"
50	Forty-sixth default program name	Character
51	Program name (first program on SLF)	Character
52	Sequence number of first program start	Integer
53	Program name (second program on SLF)	Character
54	Sequence number of second program start	Integer
"	"	"
"	"	"
"	"	"
N-1	Program name (N-50)/2 program on SLF	Character
N	Sequence number of (N-50)/2 program start	Integer

The system name and version are arbitrary identifiers. The version number is automatically incremented by 1 during a SAIL update if no other action is taken. The creation date is the date the current version was produced. Default programs are the names of the programs to be assembled by SAIL in the executive mode if no options are set to select some other set of programs. The data of each SLF can be divided into separate programs. The start of each program is assigned a sequence number which is an integral multiple of 10,000. The first program thus starts at sequence number 10,000. If the first program is less than 10,000 statements long, the second program starts at sequence number 20,000. If the first program is more than 10,000 statements long, the second program starts at the next whole multiple of 10,000 greater than the length of the first program plus 10,000.

If an updated SLF is created without resequencing the line numbers, the line numbers assigned to newly added lines consist of the line number immediately preceding their location plus a decimal part (NNNNN.XXXX). The decimal part is .X for the first 9 additional lines, .XX for additional lines 10 through 99, .XXX for additional lines 100 through 999, and .XXXX for additional lines 1000 through 9999. If more than 10,000 lines are added, the decimal part is indicated by a plus sign (NNNNN.+) when listed (either on a source code listing or a SAIL list).

The first data section on an SLF (called the prologue) starts at sequence number 1.0. The prologue (a collection of statements that may be used in more than one program) is always processed by SAIL during the executive (normal) mode.

2. OPTION DIRECTORY RECORD

The second record on an SLF (the option directory record) contains the information listed in Table A2. The option names defined in this record are used to control SAIL processing during the executive mode. Option entries can be changed, added, or deleted only in the generate, copy, or update mode. This record can not exceed 502 words (maximum 250 default options).

TABLE A2. SLF OPTION DIRECTORY RECORD

Word	Contents	Type
1	Number of lines in data portion of this SLF	Integer
2	Number of default options	Integer
3	Option one name	Character
4	Option one value	Integer
5	Option two name	Character
6	Option two value	Integer
"	"	"
"	"	"
"	"	"
N-1	Option (N-2)/2 name	Character
N	Option (N-2)/2 value	Integer

3. DATA RECORDS

The remainder of the SLF consists of data used by SAIL in producing source code files during the SAIL executive mode. These data are organized into groups of individually sequenced statements so that an input/output buffer operation can treat a large amount of data during each call. On the AFWL CRAY, SAIL defaults to 1023 words per record. Each SLF line is 11 CRAY words long, so there are 93 lines per record.

The first word on an SLF line is the sequence number. The next 9 words contain 72 characters of data, and the last word consists of the eight-character date of insertion (or last modification). Each statement can be thought of as 72 column card images. These data are of three forms: (1) SAIL executive directives; (2) SAIL maintenance directives or comments; and (3) source code input data kernels. These data kernels are output directly (or in modified form) to the source code file after being processed by SAIL executive directives.

A-II. SAIL DIRECTIVES

There are three categories of SAIL directives: (1) SAIL executive directives; (2) SAIL change directives; and (3) SAIL maintenance directives. These directives consist of up to three fields: (1) verb; (2) noun; and (3) operand. These fields are separated by one or more blanks. Table A3 lists the SAIL directives in alphabetical order, along with their types (E=Executive, C=Change, M=Maintenance).

TABLE A3. SAIL DIRECTIVES

Verb	Noun	Operand	Type
*A	Sequence number	---	C/M
*AUTO	---	---	E
*B	Program name	---	E/M
*C	Sequence number(s)	---	C/M
*C	Sequence number, column numbers	---	C/M
*D	Sequence number(s)	---	C/M
*DEFL	Option name	Logical option string	E
*DEFN	Option name	Numeric option string	E
*DIR	---	---	M
*E	---	---	E/M
*EDIR	---	---	M
*ENDPROC	---	---	E
*ETXT	---	---	E
*I	Sequence number	---	C/M
*INCLUDE	PROC name	Logical option string	E
*KEEPTO	Label name	Logical option string	E
*LABEL	Label name	---	E
*M	Sequence number(s)	---	C/M
*MAN	---	---	E

TABLE A3. CONCLUDED.

Verb	Noun	Operand	Type
*P	Subroutine name	---	M
*PROC	PROC name	Logical option string	E
*SKIPTO	Label name	Logical option string	E
*TXT	---	---	E
=	Indicates SAIL comment line	---	M

1. DIRECTIVE VERB FIELD

With the exception of the SAIL comment directive (=), all SAIL directives start with an asterisk (*) in the first character position of the record, followed immediately (no blanks) by a valid "verb." Every SAIL directive except SAIL comments has a verb field. See Table A3 for valid verbs.

In general, SAIL directives are not copied to the source code files. However, a text string not recognized as a SAIL directive (due to misspelling, etc.) will show up on the source code file. Since there is an asterisk in the first column, the line is mistakenly recognized as a comment by the CRAY FORTRAN compiler.

2. DIRECTIVE NOUN FIELD

Most executive directives require a noun field. This field consists of not more than eight nonblank characters separated from the verb field by at least one blank. For the *KEEPTO and *SKIPTO verbs, the special noun field *N, can be used where N is an integer indicating the number of lines to be retained (or skipped), instead of the usual eight-character (or less) label name.

Change directives require a sequence number in the noun field. This sequence number indicates where (in the SLF) changes are to be made.

3. DIRECTIVE OPERAND FIELD

Operand fields are usually logical. Operands are numeric, for the *DEFN verb. Numeric operands consist of option names and integer constants separated by the arithmetic operators: +, -, * (multiplication), and / (division). The resulting expression is evaluated strictly from left to right (1+2*3 = 9, not 7).

Logical operands are either true or false. A logical operand consists of one or more option names assembled in logical statements with logical relations (LT, LE, EQ, GE, GT) and logical operators (AND, OR, NOT). Integer constants can also be used, but must follow the logical relation without a delimiter (blank, comma, or equal sign) between (GT2, not GT 2). A blank logical operand field is always considered true.

Option names and operators must be separated by at least one blank. An option with a value greater than zero has a logical value of true; an option with a value of zero has a logical value of false; and an option that has not been defined will result in abnormal termination of the program. A special operator "DEF" (quotes included) can be used to test an option that may or may not be defined. The element "DEF" NAME has a value of true if the option NAME has been defined (given some integer value, including zero), or false if NAME has not been defined. Like arithmetic expressions, logical expressions are evaluated strictly from left to right unless grouped by using parentheses. Parenthetical groups are evaluated from left to right.

A-III. SAIL EXECUTIVE DIRECTIVES

1. *AUTO AND *MAN

The *AUTO directive causes automatic value and character substitution to occur until the *MAN directive is encountered. For further explanation, see Section A-VI, *Dynamic Value Substitution*.

2. *B AND *E (PROGRAM DEFINITION)

The *B NAME directive indicates the beginning of program NAME. If processing of NAME is requested, SAIL will process the PROLOGUE and skip to program NAME, as indicated by *B NAME in the SLF. Processing will terminate upon encountering *E or another *B. Also, a SAIL library listing will include the program names specified by *B directives in the summary, and will force these *B NAME lines to begin at the top of the page. Further, the program name will appear in the banner at the top of each page for all following pages, until another *B (or *E) directive is encountered.

3. *DEFL AND *DEFN (OPTION DEFINITION)

Options may be defined in three ways: (1) by the SAIL option directory record; (2) by *DEFL or *DEFN executive directives; or (3) by the SAIL input files. The SAIL option directory record is discussed in Section I2 and Section A-VI, SAIL Maintenance Directives. The SAIL input files are described in Section A-VII, SAIL Input/Output. An option value maybe changed (or defined) during executive processing by using *DEFN NAME arithmetic operand. NAME will be set equal to the arithmetic expression, evaluated strictly left to right as described above in Section A-II. An option can also be defined using the *DEFL NAME logical operand. Option NAME will have a value of zero if the logical operand expression is false, or a value of one if the expression is true.

4. *PROC, *ENDPROC, AND *INCLUDE

The first executive syntactical element used in the HULL code was the procedure definition *PROC. Its basic form is

```
*PROC  NAME  logical operand
s1
s2
"
"
"
sN
*ENDPROC
```

Statements s1 through sN (between the *PROC-*ENDPROC pair) are stored with the descriptor NAME if the logical operand is true. Statements s1 through sN may contain any data including executive directives (except for another *PROC directive). All directives contained in NAME are processed by SAIL. If the logical operand field is blank, the PROC is always created. Blank logical operand fields are always considered true.

The executive directive *INCLUDE NAME logical operand causes this block of code to be inserted at the point in the SLF where it appears (if the logical operand is true). If the procedure NAME has not been created, SAIL puts a message to that effect on output, and calls abort. Procedures can be included in the definition of other procedures using the *INCLUDE directive, up to a nesting depth of eight.

If the *ENDPROC directive is not found before encountering another *PROC (or the end of file), SAIL terminates abnormally.

Another form of the *PROC directive permits definition of a macro-like feature. In this form, the noun field is extended with a series of arguments enclosed in parentheses. For example

```
*PROC  SSQRT(A,B,C)
       "A"=SQRT("B"**2-4.0*"A"*"C")
*ENDPROC
```

where the arguments enclosed in " " are to be replaced upon expansion. To invoke the procedure defined above, the executive directive

```
*INCLUDE  SSQRT("X","Y","Z")
```

results in

```
X=SQRT(Y**2-4.0*X*Z)
```

A third form of procedure definition is also possible, by using an option table entry to define a character string for use in building a procedure name. In this usage, the noun field of the *INCLUDE directive is delimited by the character $ in the form

```
*INCLUDE  $ NAMEn AAA $  logical operand
```

where NAME is the name of a previously defined option, and n is the value of some option following NAME. If NAME has a value of M, the M options following NAME are considered to be in the "NAME option table" (see Section A-II). The first option name in the NAME option table with the value n gets concatenated with AAA (if present) to form the desired procedure name. If none of the M options in the NAME option table have a value n, SAIL ignores the directive. If the resulting procedure name has not been defined, SAIL terminates abnormally.

5. *TXT AND *ETXT

These directives instruct SAIL to ignore all directives on lines after the *TXT directive until the *ETXT directive is encountered. All lines between *TXT and *ETXT are copied to the output file as text.

6. *KEEPTO, *SKIPTO, AND *LABEL

These directives allow selective retention of blocks of code on the SLF. For short blocks (less than five lines), the form

```
*KEEPTO  *N  logical operand
```

(where N is the number of lines to be retained) causes SAIL to copy the next N lines to the source code file if the logical operand is true. The usual form for longer blocks

```
*KEEPTO NAME logical operand
```

causes SAIL to process all lines following this directive until the line

```
*LABEL NAME
```

is encountered, if the logical operand is true. If false, all lines between the *KEEPTO and *LABEL directive will be left out of the source code. The *SKIPTO directive is equivalent to the function of a *KEEPTO directive with the complement of the logical operand.

When using the *KEEPTO *N construct, SAIL comments, *P, *DIR, *EDIR, and *ETXT directives are not counted; N lines other than these will be retained or skipped as determined by the logical operand.

A-IV. SAIL CHANGE DIRECTIVES

1. *A AND *I

These change directives cause new lines to be inserted in the SLF after the sequence numbers following these directives. The syntax

```
*A N1
c1
c2
"
"
"
cN
*I N2
"
"
"
```

causes all lines after *A N1 (lines c1,c2,...,cN) to be inserted after sequence number N1. Lines on the SAIL input file following *I N2 would be inserted after sequence number N2 on the SLF.

2. *C AND *D

These directives cause changes or deletions to be made in the SLF. Records are deleted by: *C N1,N2 or *D N1,N2. All lines between N1 and N2 (inclusive) are ignored, and replaced with any lines on the SAIL input file following these directives, until SAIL finds the next change directive.

Individual columns on an SLF line may be altered by

```
*C N1(C1,C2,C1',C2')
```

which replaces columns C1 through C2 on line N1 with information from column C1' through C2' on the next SAIL input file line. Columns C1' and C2' have default values of C1'=1 and C2'=C1'+C2-C1 if missing.

3. *M

This directive is used to copy lines from one part of the SLF to another. Directive *M must be preceded by *A, *I, *C, or *D to define the insertion sequence number. The following

```
*A N1
*M N2,N3
 "
 "
 "
```

would cause lines N2 through N3 of the SLF to be inserted after sequence number N1. The *M directive can be mixed with other insertion data or with other *M directives. Lines between sequence numbers N2 and N3 are not altered by this directive, but are merely copied to the new location.

A-V. SAIL MAINTENANCE DIRECTIVES

Executive processing does not permanently change the SLF. Maintenance directives are instrumental in producing permanently changed (updated) versions of the SLF.

1. EXECUTIVE DIRECTIVES THAT ARE MAINTENANCE DIRECTIVES

The executive directives *B and *E are also maintenance directives. As such, these would be used to define the extent of new programs being added to the SLF.

2. CHANGE DIRECTIVES THAT ARE MAINTENANCE DIRECTIVES

The change directives *C, *D, *I, and *M are also maintenance directives. These directives are used to delete, insert, and rearrange lines on the SLF to form a new, permanently changed SLF.

3. *DIR, *EDIR, *P, AND SAIL COMMENTS

The *DIR and *EDIR directives aid documentation. SLF lines following a *DIR and preceding an *EDIR directive are listed during a directory list run by SAIL. Collections of FORTRAN and SAIL comment lines can then be assembled in a single listing for all or part of an SLF.

The directive *P NAME causes the SAIL listing to start on a new page, and places the subroutine name (NAME) in the banner on that page and all subsequent pages until another *P or *B (or *E) directive is encountered.

Comments which are to appear only in the SAIL listing are created by placing an equal sign (=) in the first column of each line.

A-VI. DYNAMIC VALUE SUBSTITUTION

SAIL can modify the source code it produces by using the values of options which have been defined by the option directory, the SAIL input file, and the *DEFL and *DEFN directives. Segments of the SLF can be included, left out, or repeated based on these option values. See Section A-III (SAIL Executive Directives) for further explanation.

Individual SLF lines can be altered by value or character substitution. Dynamic substitution is performed on SLF lines which begin with the character $, or on lines between the *AUTO and *MAN directives.

1. VALUE SUBSTITUTION

Current values of previously defined options are substituted for the option names on lines where dynamic substitution is to take place if they are delimited by the characters (,), or /. For example, if

```
NH=16  IMAX=60  JMAX=100
```

options have been defined, executive processing of the lines

```
$     COMMON /EOS/  RCSQ(IMAX,JMAX)
$     DATA  NH/NH/
```

results in the following lines on the source code file

```
COMMON /EOS/  RCSQ(60,100)
DATA  NH/16/
```

with the $ character removed. If the option is not defined (EOS above for example), the character string is copied intact.

Values can also be substituted for option names not delimited by the characters mentioned above, by using the special delimiter character "_" (underline). In this case, the option name and delimiters are replaced by the option value.

2. CHARACTER SUBSTITUTION

Character substitution is based on option table values as described in Section A-II. If

```
NM=3  FE=4  AL=1  AIR=6
```

options are defined, the SLF line

```
$    CALL _$NM6_(P,_NM_)
```

would be expanded to

```
CALL AIR(P,3)
```

on the source code file, with all extraneous symbols ($ and _) removed.

A-VII. SAIL INPUT/OUTPUT

The SLF described in Section A-I serves as one element of input data to SAIL during all modes of operation except an initial generate run. The SLF must have the local file name OLD when used by SAIL. When a library file is produced by a generate, copy, or update run, the new SLF will have the local file name NEW. During an executive assembly or punch run, the resulting source code will be on local file SAIL (or SAIL1, SAIL2, etc., if several programs are processed). The compiler must be instructed to use the appropriate source code file. SAIL library list runs, and messages concerning the execution of SAIL, will be put on local file OUTPUT.

Primary control of SAIL is through the local file named INPUT. This file requires the word SAIL as its first nonblank character string. Data on this file is used to determine the mode and provide change directives or other data for SAIL to complete the desired run.

The third primary input file used by SAIL is the local file INPUT2, if present. File INPUT2 is produced by program PLANK during executive assembly runs. INPUT2 supplements and overrides data on the first two records of the SLF and the INPUT file. This is the primary way for options to be defined during HULL runs.

If the default names for files INPUT and OUTPUT are not satisfactory, they may be changed at the time of execution by using

```
XSAIL I=FILE1 O=FILE2
```

where FILE1 is the local name of the SAIL input file and FILE2 is the desired name for the output file from this run.

Table A4 contains a summary of the files used or produced by program SAIL. The executable file for program SAIL is known by the local file name XSAIL on the AFWL CRAY to avoid confusion with the source code file SAIL.

TABLE A4. SAIL INPUT/OUTPUT FILES

Local file name	Purpose	Type
OLD	Old SLF	Input
NEW	New SLF	Output
SAIL	Source code file	Output
(SAIL1, SAIL2, etc.)	Source code files	Output
INPUT	Control and changes	Input
INPUT2	Alternate control	Input
OUTPUT	Lists and messages	Output

The SAIL input file generally contains all changes required for a particular run. However, long change files may be implicitly included as part of the SAIL input file by including *READ FILENAME in the input file. The asterisk (*) must be in the first column, and the file FILENAME must be a local file containing SAIL change directives. Several change files can be read using this method. Each file must be local to the job, and the SAIL input file must contain a *READ for each supplemental file.*

1. MODE PARAMETERS

Essentially, SAIL is composed of seven different programs, whose sole common feature is their use of the SLF as input, output, or both. Selection of the different modes of operation is accomplished by including a mode parameter in the SAIL input file. Table A5 shows the relationships between various files and the operational modes of SAIL.

*If more than one change file is read, the priority for any overlapping sequences in change directives goes to the earlier file read.

TABLE A5. SAIL MODE PARAMETERS AND FILE FUNCTIONS

Mode	Input files	Output files	Function	Result
EXECUTIVE (NORMAL)	INPUT INPUT2 OLD	SAIL OUTPUT	Produce source code	Source code file(s)
UPDATE	INPUT OLD	NEW OUTPUT	Produce new SLF	Updated SLF
LIST	INPUT OLD	OUTPUT	Lists SLF and attributes	Complete or partial list
COPY	INPUT OLD	NEW OUTPUT	Prepare SLF for transport	SLF in different form
GENERATE	INPUT	NEW OUTPUT	Initiate new system	New SLF
SCAN (EXTRACT)	INPUT OLD	OUTPUT	Locate text in SLF	Sequence number and lines with desired text
PUNCH	INPUT OLD	SAIL	Produce 80 column card images of SLF	Card image copy of SLF (or portion of SLF)

a. Executive--Absence of a mode parameter or the existence of a SAIL input file will cause SAIL to begin executive processing (the default mode). With no other instructions, SAIL will attempt to process the default programs using the existing option definitions on the SLF. If a SAIL input file is present, additional parameters can be used to modify the SAIL executive functions and change the resulting source code. These parameters must be located between the the initial keyword SAIL and the first input line which begins with an asterisk (a change directive or *READ to read a change file). Parameters which may be invoked during executive processing are listed in Table A6.

TABLE A6. EXECUTIVE MODE INPUT PARAMETERS

Verb	Noun(s)	Function
LINENO	---	Causes SAIL to append the SLF line numbers on the source code file
DELOPTIONS	Option name(s)	Delete options for this run
OPTIONS	Option name/ Value pairs	Add new option names and values or change values for this run
"AFTER"	Option name	Create an option table
ENDOPTIONS	---	Terminates DELOPTIONS/OPTIONS section of SAIL input
PROGRAM(S)	Program name(s)	Indicates programs to be processed for this run
ENDPROGRAM	---	Terminates program list
PROSNAME	Program name	Indicates program name for which source code is to be produced

The verbs in Table A6 can be used in other SAIL modes, but with possibly different results. Note the distinctions with care.

The LINENO parameter allows the user to establish a one-to-one correspondence between a compiler listing and an SLF listing. This simplifies program development, debug, and modification.

The DELOPTIONS/ENDOPTIONS and OPTIONS/ENDOPTIONS parameters are used to remove, add, or change options in the option directory for this run. The original option name/value pairs remain unchanged on the SLF. The SAIL input file line

SAIL DELOPTIONS A ENDOPTIONS OPTIONS B=2 C=4 ENDOPTIONS

changes

A=2 B=5 D=7

default options to

B=2 D=7 C=4

The OPTIONS parameter can also be used to establish an option table. The parameter "AFTER" (quotes included) is used in the option definition parameter list. The option name immediately following "AFTER" is the option table name. Option name/value pairs following the table name become members of the table. Therefore,

SAIL OPTIONS TAB=2 "AFTER" TAB ENZONE=3 ENZTWO=2 ENDOPTIONS

will produce the option table sequence: TAB=2 ENZONE=3 ENZTWO=2. This table exists only for this run.

The input parameters PROGRAM and PROSNAME are used to override the SLF default program definitions. Program names following the input parameter PROGRAM will be processed by SAIL. The PROSNAME parameter is used to designate the single program for which source code is to be produced during this run. Other programs (names listed after PROGRAM) are scanned for option and procedure definitions only. Program PLANK defines PROGRAM, PROSNAME, and OPTIONS on file INPUT2.

b. Update--The keyword UPDATE causes SAIL to produce a new SLF using an existing SLF and modifications specified on the SAIL input file. Table A7 lists the input parameters which may be used during update runs.

TABLE A7. UPDATE MODE INPUT PARAMETERS

Verb	Noun(s)	Function
EDIT	Character strings	Character string replacement
ENDEDIT		
LINES	Integer (le 85)	Define number of lines per page for SLF listing
SEQ	---	Resequence file NEW
NOLIST	---	Suppress listing of NEW
OPTIONS	Option name/ value pairs	Change option directory on file NEW
DELOPTIONS	Option name(s)	Delete options from option directory
ENDOPTIONS	---	Terminates OPTIONS/DELOPTIONS
PROGRAM(S)	Program name(s)	Change default programs on NEW
ENDPROGRAM	---	Terminates program list
SEQPROGRAM	Program name(s)	Resequence listed programs
VERSION	Integer	Specifies version number for NEW (or OLD)
SYSTEM	Character String	Specifies system name for NEW (or OLD)

SAIL does not process executive directives during update runs. An update run produces a modified default program list (specified by the PROGRAM parameter), a modified option directory (defined by the OPTION/ENDOPTIONS and DELOPTIONS/ENDOPTIONS parameters), and changes to the data portion of the SLF as defined by the change and maintenance directives on the SAIL input file.

The EDIT/ENDEDIT parameters are used to change character strings in the SLF. The following SAIL input lines

```
SAIL UPDATE EDIT
$DO 10$  $DO 20$
$IF(J.EQ.1)$  $IF(K.EQ.1)$
ENDEDIT
```

would cause all occurrences of DO 10 to be replaced by DO 20, and all occurrences of IF(J.EQ.1) to be replaced by IF(K.EQ.1).

c. List--The list mode produces a list of all, or a selected portion of, the SLF on the output file. Contents of the header record and the option directory record are printed first. The last segment of information printed on a list run is a summary index which lists the sequence number of each program or subroutine start (as defined by *B or *P directives). A list is always produced during an update run unless the NOLIST parameter is included.

During a list run, each program or subroutine unit starts at the top of a page with a header line identifying both program and subroutine names. Each line in the data portion consists of the sequence number, the card image, and the date this line was last modified. An asterisk precedes sequence numbers of lines modified during the last update run. The asterisk can be eliminated by including the NOAST parameter in the SAIL input file. List mode parameters are listed in Table A8.

TABLE A8. LIST MODE PARAMETERS

Verb	Noun(s)	Function
NOAST	---	Suppresses * before sequence numbers of lines changed during last update
LINES	Integer < 85	Number of lines per page on output (default 60)
PROGRAM(S)	Program Name(s)	Causes list of only those programs named
ENDPROGRAM	---	Terminates program list
DIRECTORY	---	Only lists lines between *DIR/*EDIR pairs
"DIR"	---	Modifier to allow complete listings of some programs and directory listings of others
EDIT	Character Strings	Character string replacement
ENDEDIT	---	Terminates EDIT

The SAIL input line (SAIL LIST) would produce a complete listing of the SLF. The SAIL input line (SAIL LIST PROGRAM P1 P2 P5) produces a complete listing of the programs named P1, P2, and P5. The SAIL input line (SAIL LIST DIRECTORY) produces all "directory" information (lines between *DIR and *EDIR directives). The SAIL input line (SAIL LIST DIRECTORY PROGRAM P1 P2) produces the "directory" information for programs named P1 and P2. The SAIL input line (SAIL LIST PROGRAM P1 "DIR" P2) produces the "directory" information for program P1 and a complete listing of program P2.

By including change directives after the SAIL LIST parameters, the listing incorporates the changes. Instead of new sequence numbers, the word NEW will precede all changed lines. This provides a convenient way to check the effects of changes before doing an update. The same is true of the EDIT/ENDEDIT parameters.

(d) Copy--Copy mode is similar to update except that change directives are not processed. File OLD is copied to file NEW with possible changes to the default program list (indicated by the PROGRAM parameter) and option directory (indicated by the OPTIONS, DELOPTIONS, and ENDOPTIONS parameters). The copy parameters are listed in Table A9.

TABLE A9. COPY MODE INPUT PARAMETERS

Verb	Noun(s)	Function
OPTIONS	Option name/ value pairs	Change option directory record on file NEW
DELOPTIONS	Option name(s)	Delete options from option directory
ENDOPTIONS	---	Terminates OPTIONS/ENDOPTIONS
PROGRAM(S)	Program name(s)	Change default programs on file NEW
ENDPROGRAM	---	Terminates program list
CONVERT	---	Change the file format of NEW (or OLD)
VERSION	Version number	Specifies version number for NEW (or OLD)
SYSTEM	System name	Specifies system name for NEW (or OLD)

The CONVERT parameter is order-dependent. The SAIL input line (SAIL COPY CONVERT) converts a packed internal representation file OLD to an ASCII NEW. The SAIL input line (SAIL CONVERT COPY) converts an ASCII OLD to a packed internal representation NEW. This is especially useful for transporting an SLF from one operating system to another.

e. Scan and extract--The scan mode is used to locate desired character strings in an SLF. The scan (and extract) parameters are listed in Table A10.

TABLE A10. SCAN AND EXTRACT MODE INPUT PARAMETERS

Verb	Noun(s)	Function
FIELD	Character string(s) (delimited by any character not in the character string)	Provides section of input file for desired character string(s)
ENDFIELD	---	Terminates FIELD
PROGRAM(S)	Program name(s)	Selects programs to be scanned
ENDPROGRAM	-----	Terminates program list

The extract mode is virtually identical to the scan mode, except that the resulting output file is set up as a change file. The sequence number is printed on one line, preceded by a *D change directive. The line containing the character string is printed on the next line. The user may next edit this file and make any changes to the line, and then use this file as a change file.

The SAIL input files for scan and extract

```
SAIL SCAN FIELD                SAIL EXTRACT FIELD
$character string1$            $character string1$
$character string2$            $character string2$
ENDFIELD                       ENDFIELD
```

would result in the listing of all lines containing character string1 and character string2 on the output file. For scan mode, the sequence number precedes the card image. For extract, the output file is formed as described above.

f. Punch--The punch mode produces a file consisting of 80 column-card images of the SLF without sequence numbers. The PROGRAM parameter may be used in the punch mode to select specific programs to be "punched". The SAIL input file

SAIL PUNCH PROGRAM MATLIB

would result in 80 column-card images of program MATLIB on file SAIL.

g. Generate--This is the mode used to establish a new SLF. The generate mode parameters are listed in Table A11. (Generate mode is not fully operational in AFWL's version of XSAIL in use at this printing.)

TABLE A11. GENERATE MODE INPUT PARAMETERS

Verb	Noun(s)	Function
OPTIONS	Option name/ value pairs	Establish option directory
ENDOPTIONS	---	Terminates OPTIONS
PROGRAM(S)	Program name(s)	Establish default program list
ENDPROGRAM	---	Terminates program list
SYSTEM	Name	Defines system name for SLF
VERSION	Version number	Defines version number (Default=1)

The SAIL input file

```
SAIL GENERATE
SAIL OPTIONS option1 value1 option2 value2 option3 value3
ENDOPTIONS  PROGRAM P1 P2 ENDPROGRAM  SYSTEM sysname
prologue lines
"
"
"
*B  P0
program P0 lines
"
"
"
*B  P1
program P1 lines
"
"
"
*B  P2
program P2 lines
"
"
"
*E
```

results in a new SLF on local file NEW, with the system name "sysname." The SLF is version one, and default programs are P1 and P2. The default options are option1 with value1, option2 with value2, and option3 with value3.

2. FILE PARAMETERS

The SYSTEM, VERSION, and CONVERT parameters are order-dependent file parameters.

a. SYSTEM--The SYSTEM parameter can be used to check the system name of the current SLF, or to specify the system name for the new SLF. If the parameter and its value are positioned before the copy- or update-mode parameters, SAIL checks that the system name on OLD matches the name specified; if the system name is different, SAIL terminates abnormally. If the system name matches, SAIL continues. If the SYSTEM parameter is positioned after the copy or update parameter, the specified system name is given to the revised SLF, NEW. If the SYSTEM parameter is absent, the system name on OLD is copied to NEW.

b. VERSION--The VERSION parameter can also be used in the same two ways: (1) to check the version number on OLD, or (2) to specify the version number to be assigned to NEW. If VERSION is positioned before the copy- or update-mode parameter, SAIL checks for a match between the specified number and the version number on OLD. If the numbers match, SAIL continues; if not, SAIL terminates abnormally. If VERSION is positioned after copy or update, the number specified is given to the revised SLF, NEW.

c. CONVERT--The CONVERT parameter may be used in the copy mode to convert an SLF to/from ASCII, or to/from binary. If CONVERT appears before copy, an ASCII OLD is converted to binary and copied to NEW. If CONVERT appears after copy, a binary OLD is converted to ASCII and copied to NEW.

3. EXAMPLE INPUT FILES

a. Executive--The usual SAIL input file for an executive (normal) run is structured.

```
SAIL LINENO OPTIONS option1 value1 option2 value2 ENDOPTIONS
*READ CHANG
```

where CHANG is the name of a local file containing SAIL change directives. The only required parameter for the executive mode is the word SAIL.

b. Update--The usual SAIL input file for an update run looks like

```
SAIL UPDATE OPTIONS option1 value1 option2 value2 ENDOPTIONS
*READ CHANG
```

where CHANG is the name of the local file containing the SAIL change directives that modify the SLF.

c. List--A typical SAIL input file for a list run

```
SAIL LIST PROGRAM p1
```

causes program p1 to be listed on the SAIL output file.

d. Copy--The SAIL input file

```
SAIL COPY CONVERT
```

causes SAIL to copy file OLD to file NEW. File NEW is an ASCII file.

The input file

```
SAIL CONVERT COPY
```

causes SAIL to copy an ASCII file OLD to file NEW. File NEW is a binary file.

e. Scan--The SAIL input file

```
SAIL SCAN FIELD
$do 100$
ENDFIELD
```

causes SAIL TO SCAN FILE OLD for all occurrences of do 100. All lines containing this character string get listed on output along with their sequence numbers.

f. Punch--The SAIL input file

```
SAIL PUNCH PROGRAM p1
```

causes SAIL to produce a list of program p1 on file SAIL without sequence numbers. Each line on file SAIL consists of 80 column-card images as they exist on the SLF.

g. Generate--The SAIL input file

```
SAIL GENERATE SYSTEM sysone
s1
"
"
"
sn
*B p1
ps1
"
"
"
psn
*E
```

will produce the SLF for system sysone (version one) consisting of a prologue (lines s1 through sn) and one program (p1, consisting of lines ps1 through psn). There will be no default program list and no option defaults.

APPENDIX B

AFWL HULL CODE SUMMARY

CONTENTS

CONTENTS (CONCLUDED)

B-I. HULL SYSTEM GENERATION PROCEDURES

1. FILE DESCRIPTIONS

Files used by the AFWL HULL code are listed in Table B1. Backup copies of the files stored under subdirectory /NTEPHULL/SOURCELIB are stored under subdirectory /NTEPHULL/BACKUP. Files associated with program BOW (BOWDATABASE, BUSYBIT, BOWDATA, BOWBUSY, and message files) are fully explained in Reference B1. File structures are shown in the tables that follow.

TABLE B1. AFWL HULL CODE FILES

File name	Type	CFS subdirectory location
BOWDATABASE	ASCII	/NTEPHULL/EPHULLPROBLIB
BUSYBIT	ASCII	/NTEPHULL/EPHULLPROBLIB
BOWDATA	ASCII	/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx
BOWBUSY	ASCII	/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx
HULLSTATUS	ASCII	/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx
HULLCNTRL	ASCII	/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx
T4CXXXYY	Binary	/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx
T9COOOYY	Binary	/NTEPHULL/RESTARTFILESC/PxxxxxPxxxx
HLIBA	Mixed	/NTEPHULL/SOURCELIB
Change files	ASCII	/NTEPHULL/SOURCELIB
Message files	ASCII	-- Local files only --
Temporary disk files	Binary	-- Local files only --

Files listed as Type ASCII may be examined using any text editor on the CRAY. Those listed as type binary are unformatted. To examine these files, a program must be written to read the binary file and write a new file using

---

B1. Bell, R. L., **Cray Time Sharing System (CTSS) Version of the HULL File Maintenance Program -- Bow**, AFWL-TR-84-58, Air Force Weapons Laboratory, Kirtland AFB, NM, August 1984.

formatted writes. This new file may then be examined using a text editor. A simple utility program exists under subdirectory /NTEPHULL/SOURCELIB (called XTAC) that will read restart files and write out the information for requested cells, rows, or columns as specified in its input file.

BOWDATABASE (Table B2) is an ASCII file consisting of a sequential listing of problem numbers presently being used, and the storage status of the data files (either on-line or archived). New problem numbers are inserted in sequence. The word ARCHIVE is added, if program BOW is instructed to archive a problem.

TABLE B2. BOWDATABASE FILE STRUCTURE

Record	Length	Word	Contents
1	4	1	bowdatab
		2	ase
		3	mm/dd/yy (date file was created)
		4	hh:mm:ss (time file was created)
2	4	1	xxxxx.xxxx (problem number)
		2	mm/dd/yy (date problem was added)
		3	hh:mm:ss (time problem was added)
		4	blank (or ARCHIVE)
3	4	1	xxxxx.xxxx
		2	mm/dd/yy
		3	hh:mm:ss
		4	blank (or ARCHIVE)
"	"	"	
"	"	"	
"	"	"	
N-1	4	1	xxxxx.xxxx
		2	mm/dd/yy
		3	hh:mm:ss
		4	blank (or ARCHIVE)
N	4	1	0.0
		2	LAST
		3	blank
		4	blank

BUSYBIT (Table B3) is a single-word ASCII file used to indicate that the BOWDATABASE file is "busy." The contents are assumed unique, so that a program may read the file to determine if it saved this BUSYBIT file.

TABLE B3. BUSYBIT FILE STRUCTURE

Record	Word	Contents
1	1	hh:mm:ss (current time)

The BOWDATA (Table B4) file contains a list of all restart files stored on CFS for a particular problem. If the problem contains stations, the name of the station file (T9C00000) is recorded along with the family size (if greater than one). Additional family members are stored on CFS with the names T9C000YY (where YY is the family member number between 01 and 25). The restart files have the problem time and cycle recorded with each name. Additional family members are stored on CFS with the names T4CXXXYY (where YY is the family member number between 01 and 25).

The BOWBUSY (Table B5) file is used to indicate that this problem is "busy"; that is, either a KEEL or HULL program is operating on the data files under this problem subdirectory. The last operation performed by a successful KEEL or HULL run is deletion of the BOWBUSY file.

HULLSTATUS (Table B6) is an ASCII file produced periodically by program HULL and stored under the problem subdirectory. The information is then available to monitor the progress of a calculation every ICNTRL-th cycle. The frequency of HULLSTATUS updates can be controlled by setting ICNTRL in either the HULL input file or HULLCNTRL file (ICNTRL defaults to 10).

TABLE B4. BOWDATA FILE STRUCTURE

Record	Length	Word	Contents
1	9	1	bowdata
		2	for prob
		3	xxxxx.xxxx (problem number)
		4	blank (or ARCHIVE)
		5	blank (or CYCLE or TIME)
		6	T9C00000 (or NULL)
		7	mm/dd/yy (date of last run)
		8	hh:mm:ss (time of last run)
		9	blank (or station family size)
2	4	1	T4C00000
		2	problem time (initial, usually 0.0)
		3	0.0 (problem cycle)
		4	blank (or restart file family size)
3	4	1	T4C00100
		2	problem time
		3	problem cycle
		4	blank (or restart file family size)
"	"	"	
"	"	"	
"	"	"	
N-1	4	1	T4CXXX00
		2	problem time
		3	problem cycle
		4	blank (or restart file family size)
N	4	1	LAST
		2	0.0
		3	0
		4	blank

TABLE B5. BOWBUSY FILE STRUCTURE

Record	Length	Word	Contents
1	3	1	xxxxx.xxxx (problem number)
		2	mm/dd/yy (date)
		3	hh:mm:ss (time)

TABLE B6. HULLSTATUS FILE STRUCTURE

Record	Contents
1	HULL STATUS FILE for problem xxxxx.xxxx (problem number)
1'	(possible warnings if HULLCNTRL file is not understood)
2	file created mm/dd/yy hh:mm:ss (date and time)
3	run time remaining xxx seconds
4	current priority cpr load priority lpr
5	autopriority off (or on)
5'	startday was sss today is ttt (if autopriority is on)
6	clock stop time month=mm day=dd hour=hh minute=mm
7	current value of icntrl xxx
7'	current dump interval xxx seconds (if TIMES=3)
8	current whiz estimate = xxx sec/(cell*cycle)
9	(problem header from Z-BLOCK)
10	Z-BLOCK
11-110	Z-BLOCK names and values

The HULLCNTRL (Table B7) file is an optional file that may be created by a user and stored under the problem subdirectory, to cause the running HULL program to change various parameters. If the file is found, HULL gets it, then deletes it from CFS. After reading HULLCNTRL and doing whatever is requested, HULL writes the HULLSTATUS file, stores it on CFS, and then continues. HULLSTATUS is created every ICNTRL-th cycle, whether or not file HULLCNTRL exists.

TABLE B7. HULLCNTRL FILE STRUCTURE

Record	Contents
1	problem xxxxx.xxxx (problem number)
2-N	Any of the following lines in any order
	autopriority off
	autopriority on startday ddd (mon, tue, etc)
	switch1 on
	priority pr (s, 1.0-2.0, x)
	stabf sf (0.1,...,0.99)
	timelimit runtime mm (mm in minutes)
	timelimit stoptime off
	timelimit stoptime on mon dd hh mm (mon=month, dd=day, hh=hour, mm=minute)
	control cycle xxx (new value for icntrl)
	icntrl xxx (new value for icntrl)
	dmpint xxx (sets TIMES=3 and DMPINT to xxx seconds)
N+1	end indicates last line on this file

Restart files (Table B8) are given the local file name TAPE4. Each restart file contains all the variables for a problem at a particular problem time. On CFS, each file has a unique name T4XXXYY, where XXX is the sequential number between 000 and 999, and YY is 00 (or the sequential number between 01 and 25 if the restart file is large enough to be familied). These files are unformatted, and thus cannot be read using a text editor.

The data records on the station file (Table B9) depend on the dimension and the stress option values. The header record and data record structures are shown in Tables B10 and B11. The station file is an unformatted file, and thus cannot be read using a text editor.

The initial header record (Table B10) is written by KEEL. All values beyond word 6 are set to zero. Subsequent header records are written by HULL whenever a restart file is produced. The data after word 6 are the latest

values recorded for each station. These data are used to determine if a sufficient change has occurred at each station to warrant recording new values in the active station data record (Table B11).

TABLE B8. RESTART FILE STRUCTURE

Record	Length	Word	Contents
1	4	1	555.0
		2	xxxxx.xxxx (problem number)
		3	ccc.0 (cycle number)
		4	ttt.ttttt (problem time)
2	200		Z-BLOCK
3	IMAX+JMAX+2		Mesh Coordinates
4	100*NM		Material property arrays
5	NROWPB*NVARPR		Mesh variables for first NROWPB (number of rows per block) rows. Information for each cell is pressure, velocity components, total mass, specific energy, and other variables depending on the options
5+NBLKS	NROWPB*NVARPR		Mesh variables for the last NROWPB rows. Each row has NVARPR (number of variables per row) words
Particle Records	vary		Particle numbers and coordinates (optional records)
Terminal Record	4	1	666.0
		2	666.0
		3	666.0
		4	666.0

TABLE B9. STATION FILE STRUCTURE

Record	Length	Word	Contents
1	4	1	555.0
		2	xxxxx.xxxx (problem number)
		3	0
		4	NSTN*NPP (number of stations * number of particle parameters)
2	200		Z-BLOCK
3	1024		Initial station coordinates
3'	1024		Continuation record(s) if NSTN*NPP > 1024
4	1024		Initial header record
4'	1024		Continuation record(s) if header data exceed 1024 words
5	1024		Active station data for a particular time
5'	1024		Continuation record(s) if data exceed 1024 words
"	"		"
"	"		"
"	"		"
6	1024		Restart header record (corresponding to the first restart file)
6'	1024		Continuation record(s) if header data exceed 1024 words
7	1024		Active station data for times after first restart file creation
7'	1024		Continuation record(s) if data exceed 1024 words
"	"		"
"	"		"
"	"		"

TABLE B10. STATION FILE HEADER RECORD STRUCTURE

Word	Contents
1	999.0
2	xxxxx.xxxx (problem number)
3	ccc.0 (cycle number)
4	ttt.ttttt (time)
5	NSTN (number of stations)
6	NVARST (number of variables per station)
7	PLAST(i) i=1,NSTN (last pressure recorded for each station)
7+NSTN	UCL(i) i=1,NSTN (Last U recorded for each station)
7+2*NSTN	VCL(i) i=1,NSTN (Last V recorded for each station)
[7+3*NSTN	WCL(i) i+1,NSTN (Last W recorded for each station) 3D]
	(If STRESS = 1, the last strain components are also recorded.)

2-D		3-D	
Word	*Contents*	*Word*	*Contents*
7+3*NSTN	ERRL(i) i=1,NSTN	7+4*NSTN	EXXL(i) i=1,NSTN
7+4*NSTN	EZZL(i) i=1,NSTN	7+5*NSTN	EYYL(i) i=1,NSTN
7+5*NSTN	EHOOP(i) i=1,NSTN	7+6*NSTN	EXXL(i) i=1,NSTN
7+6*NSTN	ERZL(i) i=1,NSTN	7+7*NSTN	EXYL(i) i=1,NSTN
		7+8*NSTN	EXZL(i) i=1,NSTN
		7+9*NSTN	EYZL(i) i=1,NSTN

TABLE B11. STATION FILE ACTIVE STATION DATA RECORD STRUCTURE

Word	Contents	
1	THIST	Time
2	SPERT	Number of stations at this time
n=2		
n+1	STANUM	STANUM (station number)
n+2	Material code	Material code
n+3	X coordinate	X coordinate
n+4	Y coordinate	Y coordinate
n+5	Pressure	Z coordinate
n+6	U (X velocity)	Pressure
n+7	V (Y velocity)	U (X velocity)
n+8	UDOT (X acceleration)	V (Y velocity)
n+9	VDOT (Y Acceleration)	W (Z velocity)
n+10	Density	UDOT (X acceleration)
n+11	Energy	VDOT (Y acceleration)
n+12		WDOT (W acceleration)
n+13		Density
n+14		Energy
	(If STRESS = 1, the stress deviator and strain deviator components are also recorded.)	

2-D		3-D	
Word	Contents	Word	Contents
n+12	SRR	n+15	SXX
n+13	SZZ	n+16	SYY
n+14	SHP	n+17	SZZ
n+15	SRZ	n+18	SXY
n+16	ERR	n+19	SXZ
n+17	EZZ	n+20	SYZ
n+18	EHP	n+21	EXX
n+19	ERZ	n+22	EYY
		n+23	EZZ
[n is incremented by NVARST]		n+24	EXY
		n+25	EXZ
		n+26	EYZ

The station number (STATNUM) also has the type of station coded in the fractional part of the word. The FORTRAN line: ITYP=(STANUM-INT(STANUM))*64 produces a number between 8 and 15. An Eulerian station has a value of 8. If a station is Lagrangian in the X-coordinate, the value increases by one. If a station is Lagrangian in the Y-coordinate, the value increases by two. If the station is Lagrangian in the Z-coordinate, the value increases by four. Therefore, a station that is Lagrangian in all coordinates has ITYP=15. The material code consists of a one in the bit position, for each material present in the cell containing the station. Depending on how many stations are active at a particular time, a 1024-word record may contain data for several times, or several 1024-word records may be required to record all data for a particular time. A station is considered active when at least one of the following variables has changed by more than one percent of its last recorded value: pressure, velocity component, density, or energy.

HLIBA (Table B12) is a collection of executable, source, and relocatable code. HLIBA was created to eliminate the need to get multiple files from CFS. Once HLIBA is local, the command line: LIB HLIBA<esc>X ALL.<esc>END<carriage return> will extract the above files and leave them in the local file space, where they can be used as required.

Change files (Table B13) are ASCII files, containing SAIL change directives followed by revised source code. See Appendix A (AFWL SAIL Code Summary) of this report (AFWL Vectorized EPHULL Code User Manual) for a complete explanation.

TABLE B12. HLIBA FILE STRUCTURE

Record	Contents
1	OLD (the current HULL SLF)
2	HULLIB (relocatable utility routines)
3	MATLIB (ASCII material property library)
4	XSAIL (executable program SAIL)
5	BOW (executable program BOW)
6	PLANK (executable program PLANK)

TABLE B13. CHANGE FILE STRUCTURE

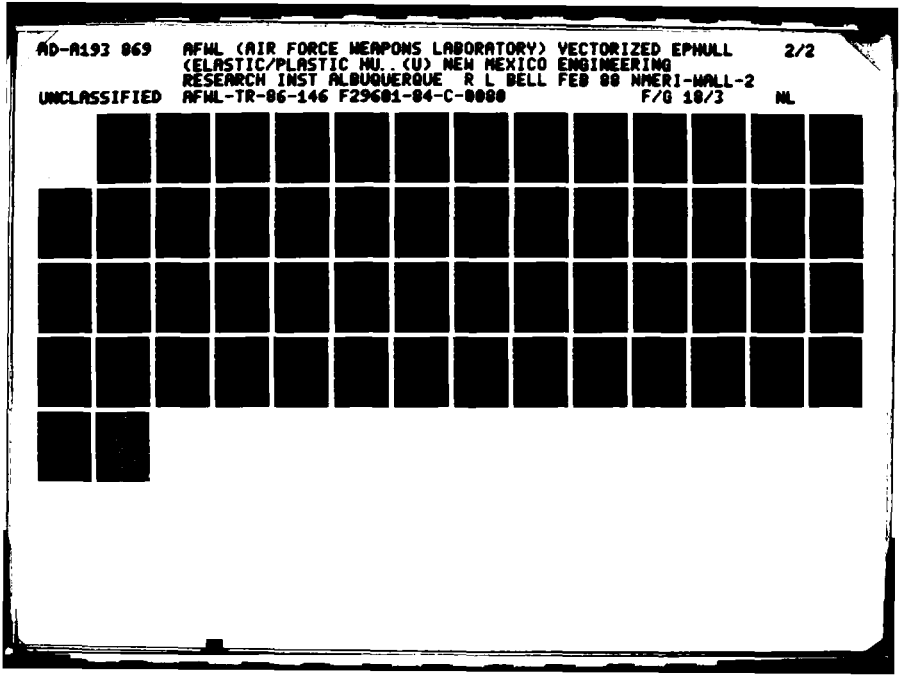
Record	Contents
1	SAIL change directive
2	Revised source code
3	SAIL change directive
4	Revised source code
"	"
"	"
"	"

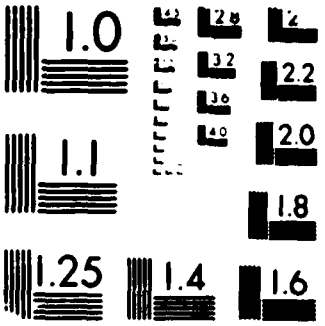
Four message files (Table B14) are created by program BOW, and left in the job file space for use by other programs. The KEELMSG file notifies KEEL that BOW has added the appropriate CFS subdirectory for this problem. KEEL uses KEELMSG as the initial BOWDATA file. The HULLMSG file is used by HULL to determine the current restart file name, so that subsequent restart files will have the appropriate names on CFS. The PULLMSG file for program STATION only contains the station file name, and lets program STATION know that the station file is available on CFS. For program PULL, PULLMSG contains the names of all restart files for which plots have been requested.

TABLE B14. MESSAGE FILE STRUCTURE

Record	Length	Word	Contents
1	9	1	bowdata
		2	for prob
		3	xxxxx.xxxx (problem number)
		4	blank (or ARCHIVE)
		5	NULL (or T9C00000)
		5	blank (or CYCLE or TIME)
		6	NULL (or T9C00000)
		7	mm/dd/yy
		8	hh:mm:ss
		9	blank (or station family size)
2	4	1	T4CXXXYY
		2	ttt.ttttt (problem time)
		3	ccc (problem cycle)
		4	blank (or restart file family size)
2'	4		Additional restart file names if message is for PULL and more than one restart file is being plotted.
3	4	1	LAST
		2	0.0
		3	0
		4	blank

All mesh variables need not be in core all the time. Depending on the options for a particular calculation, as few as three rows may be required in core to process the calculation. If running a calculation in-core, the files listed in Table B15 are not created. If running in the disk mode, the mesh variables for the entire mesh are transferred (one row per record) from the appropriate restart file (local TAPE4) to the random disk file TAPE23. As rows are initially processed, they are read from Diskc (unit 23, TAPE23) into central memory, then written out to Diskb (unit 22, TAPE22). After all rows have been processed, the unit numbers for Diska (unit 21) and Diskb (unit 22) are swapped. The last row written on Diskb is recorded in variable LAST. During subsequent cycles, rows are read from Diska until the row number in


AD-A193 869
AFWL (AIR FORCE WEAPONS LABORATORY) VECTORIZED EPHULL
(ELASTIC/PLASTIC HU..(U) NEW MEXICO ENGINEERING
RESEARCH INST ALBUQUERQUE R L BELL FEB 88 NMERI-WALL-2
2/2
UNCLASSIFIED
AFWL-TR-86-146 F29601-84-C-0080
F/G 18/3
NL



MICROCOPY RESOLUTION TEST CHART

LIMB. Rows above LIMB are read from Diskc. Once LIMB equals JMAX (or KMAX, in 3-D), the entire mesh is on TAPE21 and TAPE22, so there is no need to keep TAPE23. TAPE23 is thus destroyed, to reduce the number of files being charged to the job.

TABLE B15. TEMPORARY DISK FILES

File	Contents
TAPE21	Active mesh (one row per record)
TAPE22	Active mesh
TAPE23	Full mesh

2. COSMOS FILES FOR SYSTEM GENERATION

Copies of all files required for system generation are located under subdirectory /NTEPHULL/SYSGEN.

a. Libgen--The COSMOS file LIBGEN (Table B16) consists of the appropriate MASS commands, SAIL input files, CFT, and LDR instructions to produce the files needed for Composite HULL.

TABLE B16. LIBGEN COSMOS FILE

```
*INTERRUPT ON SOFTWAREERROR TO EXIT
*MASS GET DIR=/NTEPHULL/SYSGEN OLD:EPHULL120 XSAIL AFWLCH VPROC
*FILE NAME=INPUT,DUPLICATE=DESTROY,END=EOR
SAIL LINENO OPTIONS OBJLIB=0 ENDOPTIONS
PROGRAM PLOTTERS LIBRARY
*READ AFWLCH
*READ VPROC
EOR
*XSAIL I=INPUT O=LIBSAIL
*CFT I=SAIL,B=BLIB,ON=DINXZ,L=LIBLST
*CFT 1=SAIL1,B=BPLOT,ON=DINXZ,L=PLTLST
*BUILD NL=HULLIB,B=(BLIB,BPLOT),LOZMXL=LIBMAP
*DESTROY ALWITH. SA
```

TABLE B16. CONCLUDED.

```
*FILE NAME=INPUT,DUPLICATE=DESTROY,END=EOR
SAIL LINENO OPTIONS VECTOR=1 ENDOPTIONS
PROGRAM BOW PLANK EOS
*READ AFWLCH
*READ VPROC
EOR
*XSAIL I=INPUT O=BPSAIL
*CFT I=SAIL2,B=BPLNK,ON=DINXZ,L=PLNKLST
*CFT I=SAIL1,B=BBOW,ON=DINXZ,L=BOWLST
*LDR BIN=BBOW,LIB=(HULLIB,CFTMATH),MO=FULL,ML=BOWMAP
*LDR BIN=BPLNK,LIB=(HULLIB,CFTMATH),MO=FULL,ML=PLNKMAP
*DESTROY ALWITH. SA
*FILE NAME=INPUT,DUPLICATE=DESTROY,END=EOR
SAIL PUNCH PROGRAM MATLIB
*READ AFWLCH
*READ VPROC
EOR
*XSAIL I=INPUT O=MATSAIL
*SWITCH SAIL MATLIB
*LIB HLIBA NEWFILE
ADD OLD HULLIB MATLIB XSAIL BOW PLANK
END
*MASS STORE /NTEPHULL/SOURCELIB/HLIBA
*EXIT:
*CONCAT SAILOUT/CC LIBSAIL/CC BPSAIL/CC MATSAIL/CC
*DISPOSE DN=SAILOUT,SF=CC,FID=filename1
*CONCAT LIBLIST LIBLST LIBMAP
*CONCAT PLNKLIST PLNKLST PLNKMAP
*CONCAT BOWLIST BOWLST BOWMAP
*DISPOSE DN=LIBLIST,FID=filename2
*DISPOSE DN=PLNKLIST,FID=filename3
*DISPOSE DN=BOWLIST,FID=filename4
*SELECT PRINTLOG=LIBLOG
*DISPOSE DN=LIBLOG,FID=filename5
```

The LIBGEN file may be processed interactively by entering

COSMOS LIBGEN / time priority <carriage return>

or it may be submitted by entering

SUBMIT LIBGEN / time priority

b. BOWDATABASE Initialization--The COSMOS job stream shown in Table B17 can be run after LIBGEN. The executable program BOW adds subdirectories /NTEPHULL/RESTARTFILESC and /NTEPHULL/EPHULLPROBLIB. A new blank BOWDATABASE file is created and stored under /NTEPHULL/EPHULLPROBLIB.

CAUTION: running this job stream on a system where a current BOWDATABASE exists will erase all information on that file.

TABLE B17. BOWGEN COSMOS FILE

```
*FILE NAME=BOWIN,END=EOR
BOW CREATE
EOR
*BOW I=BOWIN O=BOWOUT
*SELECT PRINTLOG=BOWLOG
```

B-II. HULL SYSTEM OPERATION PROCEDURES

The following programs are generally executed as part of a COSMOS run stream. Occasionally they may be executed interactively, to track down a coding error or to accomplish some special purpose. Options are listed in the next section. The following is a summary description of each program.

1. PROGRAM BOW

Program BOW is used to keep track of the problem numbers and files associated with each problem.

a. KEEL--For a mesh initialization run, BOW compares the problem number with the numbers recorded on BOWDATABASE. If the number is not found there, it is inserted in sequence on BOWDATABASE, and BOW adds the appropriate subdirectory /NTEPHULL/RESTARTFILESC/PxxxxxPxxxx. If the problem number already exists on BOWDATABASE and the RERUN parameter is not included in the KEEL input file, BOW aborts. For a new problem number (or an old one with RERUN specified), BOW attempts to save the BOWBUSY file under the problem subdirectory. If the save fails, BOW calls abort. After a successful save for a problem being rerun, BOW gets file BOWDATA and deletes all files listed.

b. HULL--For a HULL run, BOW attempts to save BOWBUSY under the problem subdirectory. If the save fails, BOW calls abort. If the save is successful, BOW gets BOWDATA and compares the cycles (and times) for each listed restart file with the restart instructions on the HULL input file. BOW gets the appropriate restart file (and station file if required) and deletes all subsequent restart files.

For KEEL and HULL runs, BOW leaves the BOWBUSY file under the problem subdirectory, to assure that no other job has modify access to the data files until the KEEL or HULL job finishes. The last action taken by KEEL and HULL is to delete BOWBUSY.

c. PULL and STATION--For a PULL or STATION run, BOW does not attempt to save BOWBUSY, since no files under the problem subdirectory are to be modified. For PULL, BOW gets BOWDATA and compares plot request times (or cycles) on the PULL input file with restart times (cycles) on BOWDATA. BOW creates a file

(PULLMSG) containing the names of the restart files for which plots are required. For STATION, BOW gets BOWDATA to determine if a station file exists for this problem, and creates a file (also known as PULLMSG) containing the station file name. BOW gets the first restart file (or the station file) from CFS with local file name TAPE4, writes a summary message on its output file, and then terminates.

d. BOW--Program BOW may also be executed independently, to accomplish a multitude of file maintenance actions (Ref. B1). The most usual actions are: (1) archive problems; (2) retrieve problems; (3) remove problems; (4) list problems; and (5) edit. Table B18 lists BOW parameters.

TABLE B18. BOW PARAMETERS

Parameter	Function
BOW	Indicates file maintenance actions are required.
CREATE	Adds root nodes and major subdirectories on CFS; writes and stores a blank BOWDATABASE file under subdirectory /NTEPHULL/EPHULLPROBLIB.
ADD	Adds problem number to BOWDATABASE.
REMOVE	Deletes all files associated with the listed problem numbers from CFS. Removes problem information from BOWDATABASE. Removes problem subdirectories from CFS.
MODIFY	Changes CFS file information on BOWDATA file.
ADD	Adds restart (or station) files to BOWDATA.
DELETE	Deletes listed restart (or station) files from BOWDATA and CFS.
CHANGE	Changes information about particular restart file (time or cycle).
ARCHIVE	Changes the CFS USE parameter for all files for all problems listed to USE=A (archive). Adds word ARCHIVE to records in BOWDATA and BOWDATABASE.
RETRIEVE	Changes the CFS USE parameter for all files for all problems listed to USE=X (weekly). Deletes word ARCHIVE from records in BOWDATA and BOWDATABASE
EDIT	Lists BOWDATABASE on output.
LIST	Lists BOWDATA for requested problem numbers on output.
ALL	Lists BOWDATA for all problem numbers (on BOWDATABASE) on output.
KEEL	Indicates that a problem is being initialized.
RERUN	Indicates that this problem number may already be listed on BOWDATABASE. If found, all files under the subdirectory will be deleted.
PROB or PROBLEM	Indicates that the next word on the input file will be the problem number.
HULL	Indicates that a problem is being continued.
PROB or PROBLEM	Same as for KEEL.
T or TIME	If found before the word INPUT, indicates requested restart time (consecutive TIME restarts not allowed).
C or CYCLE	If found before the word INPUT, indicates requested restart cycle (consecutive CYCLE restarts not allowed). (If both TIME and CYCLE are specified, BOW does not object to consecutive TIME or CYCLE restarts).

TABLE B18. CONCLUDED.

Parameter	Function
INPUT	Terminates restart field. TIME or CYCLE following this word changes the ZBLOCK values of T and CYCLE.
PULL	Indicates that a problem is being plotted.
STATION	Indicates that station data are to be plotted.
PROB or PROBLEM	Same as for KEEL.
FTIME	First time to be plotted (default, time on T4C00000).
LTIME	Last time to be plotted (default, time on last restart file).
CTIME	Plot only for times listed after this word.
CCYCLE	Plot only for cycles listed after this word.

The input file BOW ARCHIVE PROB 1.1 causes all files listed on BOWDATA for problem 1.1 to be modified to "USE=A" on CFS, and the word ARCHIVE to be added to the problem 1.1 record on BOWDATABASE and to BOWDATA for problem 1.1. The input file BOW RETRIEVE PROB 2.2 causes all files listed on BOWDATA for problem 2.2 to be modified to "USE=X" on CFS, and the word ARCHIVE to be removed from BOWDATA and the problem 2.2 record on the BOWDATABASE file. The input file BOW REMOVE PROB 1.2 causes BOW to delete all files listed on BOWDATA for problem 1.2 from CFS, and to remove the problem 1.2 subdirectory. The data record for problem 1.2 also is deleted from the BOWDATABASE file. The input file BOW LIST PROB 1.3 causes BOW to copy the BOWDATA file for problem 1.3 to output. The input file BOW EDIT causes BOW to copy the BOWDATABASE file to output.

2. PROGRAM PLANK

This program is seldom used independently. Its primary function is to provide the alternate input file (INPUT2) for program SAIL. For a KEEL run, program PLANK reads the options set in the KEEL input file and writes them out to file INPUT2. For a HULL, PULL, or STATION run, program PLANK reads the options set in the appropriate TAPE4 Z-BLOCK (and any additions or changes from

the appropriate input file), and then writes them out to file INPUT2. The Z-BLOCK options are listed in Table B19. Additional options which have meaning to program PLANK are listed in Tables B20 through B23.

TABLE B19. Z-BLOCK OPTIONS

Option	Function	Program
PROB	Problem number	K
AREF	Aft reflective (3-D)	K, H
ATMOS	Type of ambient atmosphere (default, 2) 1. Tropical 2. Temperate 3. Arctic 4. Exponential 5. Constant	K
BREF	Bottom reflective (default, true)	K, H
BURN	HE detonation (default, 0, no detonation) 1. Physical burn 2. Programmed burn	K, H
CYCLE	Number of computation cycles completed	K, H
DIMEN	Number of dimensions in problem (default, 2) 1. One dimension (not checked out) 2. Two dimensions (default) 3. Three dimensions	K
DT	Current time increment in seconds	K, H
DVISC	Deviatoric viscosity option (default, 0) 1. Option 1 2. Option 2	K, H
HELC	Energy last cycle (ergs)	
ETH	Theoretical energy in mesh (ergs)	
EXPAND	Fraction of old mesh to be added to new upon rezone	K, H
FAIL	Material failure model (default, 0, no failure) 1. Ultimate stress or strain 2. Accumulated strain	K, H*
FBLOS	Nuclear fireball cooling (default, 0) 1. Code included	K, H

TABLE B19. CONTINUED.

Option	Function	Program
FLUXER	Euler advection method 1. Single material--no stress 2. Not supported 3. Multimaterial (diffusion limited)	K
FREF	Front reflective (3-D)	K, H
GEOM	Geometry 1. Cartesian (default for 3-D) 2. Cylindrical (default for 2-D) 3. Spherical (not checked out)	K, H
HOB	Height of burst above sea level (km)	K
IMAX	Number of cells in the X-direction	K
IQ	First quiet column in the X-direction (calculations only accomplished for I=1 to IQ)	K, H
ISLAND	Reflective cell option (default, 0) 1. Infinite mass 2. Finite mass	K, H
JMAX	Number of cells in the Y-direction	K
JQ	First quiet row in the Y-direction (calculations only accomplished for J=1 to JQ)	K, H
KMAX	Number of cells in the Z-direction	K
KQ	First quiet plane in the Z-direction (calculations only accomplished for K=1 to KQ)	K, H
LAGRAN	Lagrangian grid option (default, 0) 1. Interactive Lagrange module 2. Pure Lagrange module	K
LREF	Left reflective (2-D default, true)	K, H
MAGFLD	Geomagnetic field option (default, 0) 1. Code included	K
METHOD	Differencing scheme option (default, 2) 1. SHELL I 2. SHELL II	K, H
MLC	Mass last cycle (g)	
MTH	Theoretical mass (g)	
NH	Number of variables per cell	

TABLE B19. CONTINUED.

Option	Function	Program
NHIC	Number of mesh variables in core	
NHIST	Number of hysteretic variables advected with mass	
NM	Number of materials in the problem	K
NOP	Number of particles in the problem	K
NPLPB	Number of planes per block (3-D)	
NPP	Number of particle parameters	
NROWPB	Number of rows per block (2-D)	
NSTN	Number of stations (in Eulerian mesh)	K
NVARST	Number of variables per station	
PTSTOP	Problem time stop (s)	K, H
RAD	Radiation option (default, 0) 1. Equilibrium (not checked out) 2. Nonequilibrium (not checked out)	K, H*
RADLOS	Theoretical energy radiated	
REZONE	Rezone option (default, 0, no rezone) 1. Standard shock 2. Fireball follower 3. Horizontal shock follower 4. Vertical shock follower 5. Particle follower 6. Particle follower, shocks ignored 7. Continous rezone 8. Vertical shock follower (same as 4)	K, H
RIGMAS	Rigid body mass for ISLAND=2	
RIGVEL	Rigid body velocity for ISLAND=2	
RREF	Right reflective (2-D default, false)	K, H
SPIN	Spinning axisymmetric coordinate system	K
STABF	Stability factor for Courant condition time step	K, H
STRAIN	Strain option (default, 0) 1. Keep strain deviators	K, H*
STRESS	Stress option (default, 0) 1. Keep stress deviators	K, H*
SUME	Cumulative radiation energy loss	
T	Problem time in (s)	K, H
TERAD	Cumulative energy radiated	

TABLE B19. CONCLUDED.

Option	Function	Program
TLC	Time last cycle	
TREF	Top reflective (default, false)	K, H
TTIME	Total computer time (s)	
TTSTOP	Total allowable computation time (h)	K, H
UREZ	X-direction rezone velocity (cm/s)	K, H
VISC	Artificial viscosity option (default, 0) 1. Code included	K, H
VOIDS	Explicit void option (default, 0) 1. Code included	K, H*
VREZ	Y-direction rezone velocity	K, H
WORK	Plastic work hardening option (default, 0) 1. Code included	K, H*
WREZ	Z-direction rezone velocity	K, H
XOB	X-coordinate of burst (km)	K
YGND	Altitude of ground above sea level (cm)	K
YOB	Y-coordinate of burst (km) (3-D)	K
YIELD	Yield of burst (kt)	K

Note: Under program heading: K = May be set in KEEL
H = May be set in HULL
H* = May be reset in HULL if originally set in KEEL

TABLE B20. PLANK PROGRAM PARAMETERS

Parameter	Function
KEEL, HULL, PULL	Indicates program to be produced

TABLE B21. PLANK PARAMETERS FOR KEEL

Parameter	Function
KEEL	Program to be produced
PROB or	Next word will be
PROBLEM	Problem number
HEADER	Next input line contains problem title
MESH	Indicates end of option section on input file
FIREIN	Option to initialize mesh, using information from previous calculation
ENDFIRE	Terminates FIREIN sequence
END	Indicates end of input file
LAGRANGE	Indicates Lagrangian mesh

TABLE B22. PLANK PARAMETERS FOR HULL

Parameter	Function
HULL	Program to be produced
PROB or PROBLEM	Next word will be the problem number
CYCLE	Restart cycle (if before INPUT) or specify CYCLE
T	Restart time (if before INPUT) or specify T
TTIME	Specify TTIME
TTSTOP	Specify TTSTOP
PTSTOP	Specify PTSTOP
CSTOP	Specify CSTOP
INPUT	Terminate restart section of input file
END	Terminate input file
VECTOR	Indicates that VECTOR code is to be produced
NOVECT	Indicates that scalar code is to be produced (default)
SWITCHS	
ON	Sets option BUFFER=2 for double buffering
OFF	Sets option BUFFER=1 for single buffering (default)
DOUBLE	Sets option BUFFER=2 for double buffering
TEKCOLOR	Sets option COLOUR (default, 0) 1. Half plane 2. Mirrored 3. Split

TABLE B23. PLANK PARAMETERS FOR PULL

Parameter	Function
PULL	Program to be produced
STATION	Indicates that program STATION is to be produced
PROB or PROBLEM	Next word will be the problem number
HEADER	Indicates that the next line on the input file contains a new problem title
END	Terminates the input file
TEKCOLOR	Sets option COLOUR (default, 0) 1. Half plane 2. Mirrored 3. Split

Tables B24-B26 pertain to the recently implemented Lagrangian capability. These tables are provided for information, but should not be considered either reliable or complete. Users of the Lagrangian capability should refer to HULL *Documentation, Volumes I and II (Ref. B2).*

B2. Matuska, D. A., and Osborn, J., **HULL Documentation**, Orlando Technology, Inc.

TABLE B24. LAGRANGIAN OPTIONS

Option	Function/value
NHIL	Max array size for element variables
NHIN	Max array size for node variables
NOMAT	Number of materials
NMMAX	Max material number
NLREG	Number of Lagrangian regions
LINKH	Link option (default, 0, no link) 1. Interlinked 2. Velocity link 3. Stress link
LTIME	Number of Lagrangian stations (in Lagrangian mesh)
LSTRAN	GT 0 indicates that strains are to be recorded
NGRID	
NTTYPE	Max number of node-type definition lines (50)
NMATC	Max number of material definition lines (50)
NSTYPE	Max number of shape lines (50)
NDROP	
NSTNH	
NINT	
NLBOUND	Max number of elements on a boundary (slide or linked)
INMEML	Flag to indicate if all elements are in memory 0. LECM elements in memory 1. All elements in memory
INSAV	
LRIGID	Max value for RIGIDL for all regions
LSLIDE	Slide line option 0. No slide lines 1. Slide lines in problem
NLNODE	Max number of slave nodes for any region
LAIRB	Lagrangian air option for explosive problems
NPRES	GT 0 indicates pressure nodes
IRZCON	
NHILR	

TABLE B25. LAGRANGIAN REGIONAL PARAMETERS

Parameter	Function
LAGRANGE	Indicates Lagrangian portion of input
PROB	Next word will be the problem number
HEADER	Next input line will be the problem title
STRAINL	Nonzero indicates that element strains are to be saved
GEOM	Sets option LGEOM (default, 0) 1. Cartesian 2. Axisymmetric (about Y-axis)
LECM	Number of elements in memory at any time (default, all)
NM	Sets options NOMAT and NMMAX
REGION	Indicates region number
NXI	Number of nodes in X-direction
NYJ	Number of nodes in Y-direction
NT	Node type
INT	Interlinked (sets option LINKH=1)
STRESS	Stress link (sets option LINKH=3)
SLAVE	Slave (sets option LSIDE=1)
SLAVE1	Slave (sets option LSLIDE=1)
SLAVE2	Slave (sets option LSLIDE=1)
SLAVE3	Slave (sets option LSLIDE=1)
LOCKED	Slave (sets option LSLIDE=1)
PRES	Pressure (sets option NPRES=1)
AND, FOR	Skipped
MASTREG	Master region number
NX	Node numbers along boundary { set Options
NY	Node numbers along boundary { NLBOUND and NLNODE
RIGIDL	Sets option LRIGID to max value

TABLE B26. LAGRANGIAN MESH STATION PARAMETERS

Parameter	Function
STATIONS	Indicates that stations will be generated
XL	X-coordinate of station
YL	Y-coordinate of station Existence of (XL,YL) pair causes number of stations (LTIME) to be incremented

3. PROGRAM SAIL

The executable code for program SAIL is called XSAIL. See Appendix A (AFWL SAIL Code Summary) of this report (AFWL Vectorized EPHULL Code User Manual) for the XSAIL modes of operation. The primary purpose for SAIL in the HULL system is to maintain and manipulate the SAIL library file for the Composite HULL code.

4. PROGRAM KEEL

This program sets up the initial restart file (and station file, if required). The KEEL input file contains the options desired for a particular problem, the size and physical extent of the mesh, the geometries for various materials in the problem, and the initial locations of stations (if included).

5. PROGRAM HULL

This program integrates a problem through time. The values of various parameters in the HULL input file determine when restart files are produced and when the problem is considered finished.

6. PROGRAM PULL

This program provides graphic presentation of the values recorded in each restart file. Various options are available to provide split and overlaid plots. PULL plots data for the whole mesh as a function of location, for a particular time.

7. PROGRAM STATION

Like PULL, STATION produces graphic output. STATION provides data as a function of time for a particular location (station) in space.

B-III. SUMMARY OF HULL SYSTEM OPTIONS

1. KEEL

The Z-BLOCK options listed in Table B19 (with K under the program heading) may appear in a KEEL input file before the keyword MESH. The complete input line following the line containing the keyword HEADER is used as the problem title. All information on the line after the word HEADER will be ignored. A KEEL input file should conform to the format

```
KEEL (RERUN) PROB xxxxx.xxxx
ZBLOCK option list
HEADER
Problem title card
MESH
Mesh generation instructions
GENERATE
Material packaging instructions
(ISLAND packaging instructions)
(STATION creation instructions)
END
```

where items in parentheses are optional. Additional parameters recognized by program KEEL are listed in Tables B27 through B35.

TABLE B27. BASIC KEEL PARAMETERS

Parameter	Function
KEEL	Identifies KEEL input file
RERUN	Used by program BOW
PROB or PROBLEM	Next word is problem number
HEADER	Next input line is problem title
MESH	Terminates ZBLOCK option section of input file
GENERATE	Terminates mesh section of input file
END	Terminates KEEL input file

TABLE B28. KEEL STANDARD MESH PARAMETERS

Parameter	Function
MESH	Identifies mesh construction section of input
XO	Minimum X-coordinate of mesh
XMAX	Maximum X-coordinate of mesh
YO	Minimum Y-coordinate of mesh
YMAX	Maximum Y-coordinate of mesh
ZO	Minimum Z-coordinate of mesh (3-D)
ZMAX	Maximum Z-coordinate of mesh (3-D)
DX, DY, (DZ)	Size of cells in each coordinate
NX, NY, (NZ)	Number of cells of size given above Defaults: NX=IMAX, DX=(XMAX-XO)/IMAX, etc.
CONSTANT SUBGRID	Notifies KEEL that a constant subgrid is desired

TABLE B29. KEEL SUBGRID MESH PARAMETERS

Parameter	Function
CONSTANT SUBGRID	Notifies KEEL that a constant subgrid is desired
XO, YO, (ZO)	Minimum subgrid coordinates
XMAX, YMAX, (ZMAX)	Maximum subgrid coordinates
NX, NY, (NZ)	Number of cells in subgrid (for each coordinate)
XOLIMIT, YOLIMIT, (ZOLIMIT)	Minimum mesh coordinates
XMAXLIMIT, YMAXLIMIT, (ZMAXLIMIT)	Maximum mesh coordinates
RXNEG, RYNEG, (RZNEG)	Maximum expansion rates in negative coordinate directions
RXPOS, RYPOS, (RZPOS)	Maximum expansion rates in positive coordinate directions
NXL, NYB, (NZB)	Number of cells "below" subgrid
NXR, NYT, (NZT)	Number of cells "above" subgrid

TABLE B30. KEEL GENERATE PARAMETERS

Parameter	Function
GENERATE	Identifies the geometry section of input
ISOENERGY	Identifies default geometry for hot sphere of specified (kt) yield
Name	Identifies name of material to be placed in the mesh
RHO	Density (default=ambient density in g/cm^3)
I	Specific internal energy (default=ambient energy in ergs/g)
U	X-velocity (default, 0.0 cm/s)
V	Y-velocity (default, 0.0 cm/s)
W	Z-velocity (default, 0.0 cm/s) (3-D)
FIREIN	Material properties determined by previous calculation
HULL	HULL calculation
FILE	File name for previous calculation restart file
SPUTTER	SPUTTER calculation
FB N	Fireball number
SAP	SAP calculation
FILE	File name containing SAP results
SCALE	Scale input data to new yield
YIELDIN	Yield of input data (kt)
Geometry	See Tables B31 and B32
ENDFIRE	Terminates FIREIN sequence
PARTICLES	Indicates tracer particles are to be placed in the mesh
PRAD	Radius of sphere centered at HOB to contain particles
Geometry	Geometry to contain particles (default, sphere)
PACKAGE	Very flexible mesh filling syntax
Geometry	See Tables B31 and B32
DELETE	Exception to PACKAGE
Geometry	Geometry where material is not to be packaged
ISLAND Geometry	Geometry of reflective cells
STATIONS	Indicates that stations are to be generated
XL, YL, (ZL, 3-D)	Lagrangian station coordinates
XS, YS, (ZS, 3-D)	Eulerian station coordinates (a station may have some Lagrangian and some Eulerian coordinates)

TABLE B31. TWO-DIMENSIONAL GEOMETRIES FOR KEEL

Geometry	Parameters
XL, YL, (ZL, 3-D)	Lagrangian station coordinates
XS, YS, (ZS, 3-D)	Eulerian station coordinates (a station may have some Lagrangian and some Eulerian coordinates)
RECTANGLE or RECTAROT	XL, XR, YB, YT (extent of rectangle)
TRIANGLE or TRIROT	X1,Y1, X2,Y2, X3,Y3 (coordinates of vertices)
CIRCLE	XC, YC, RAD (coordinates of center, radius)
ELLIPSE	A, B, C, D (where (Y-A)**2/B**2+(X-C)**2/D**2=1.0), or XR, YR (where XR and YR are intercepts on the X- and Y-axis)
PARABOLA or PARABLROT	A, B, C (where Y-A=B*(X-C)**2), or YR, XR (where YR and XR are intercepts on the Y- and X-axis)
HYPERBOLA or HYPEROT	A, B, C, D (where (Y-A)**2/B**2-(X-C)**2/D**2=1.0)
GNRLFIT	A, B, C, D, E, F (where Y=AX**5+BX**4+CX**3+DX**2+EX+F)
CURVE TABLE	
CURVE TABLE NPT NN	(X1,Y1),(X2,Y2),(X3,Y3),...,(XNN,YNN) NN < 101

TABLE B32. THREE-DIMENSIONAL GEOMETRIES FOR KEEL

Geometry	Parameters
BOX	XL, XR, YB, YF, ZB, ZT (extent of box)
WEDGE	X1,Y1, X2,Y2, X3,Y3, ZB,ZT (extent of wedge)
PYRAMID	X1,Y1,Z1, X2,Y2,Z2, X3,Y3,Z3, X4,Y4,Z4 (tetrahedron)
CYLINDER	XC, YC, ZB, ZT, RADIUS
ELLIPCYL	A, B, C, D, ZB, ZT (where A, B, C, D are coefficients for an ellipse), or XR, YF, ZB, ZT (where XR and YF are intercepts on X- and Y-axis)
PARACYL	A, B, C, ZB, ZT (where A, B, C are coefficients for a parabola), or XR, YF, ZB, ZT (where XR and YF are intercepts on X- and Y-axis)
HYPERCYL	A, B, C, D, ZB, ZT (where A, B, C, D are coefficients for a hyperbola)
SPHERE	XC, YC, ZC, RADIUS
ELLIPSOID	AY, BY, CY, DY, AZ, BZ, CZ, DZ (where these are coefficients for ellipses in the XY and XZ planes) (note: AY=AZ=0 and CY+DY=CZ+DZ), or XR, YF, ZT (where these are axis intercepts)
PARABALOID	A, B, C, Y1, Z1 (where A, B, C are coefficients for the parabola and (Y1,Z1) is max point of the intercept with the plane Y=Y1), or XR, YB, ZT (where these are axis intercepts)
HYPERLOID	AY, BY, CY, DY, AZ, BZ, CZ, DZ (where these are coefficients for a hyperbola in the XY and XZ planes)
HYPERPARA	AY, BY, CY, DY, AZ, BZ, CZ, DZ (where these are coefficients for a parabola in one plane and a hyperbola in the other)
RECTAROT	XL, XR, ZB, ZT (rectangle rotated about the Z-axis)
TRIROT	X1,Z1, X2,Z2, X3,Z3 (triangle rotated about the Z-axis)
CIRCLEROT	XC, ZC, RAD (circle rotated about the Z-axis)
ELLIPSROT	AZ, BZ, CZ, DZ (ellipse rotated about the Z-axis), or XR, ZT (where these are axis intercepts)
PARABLROT	AZ, BZ, CZ (parabola rotated about the Z-axis), or XR, ZT (where these are axis intercepts)
HYPEROT	AZ, BZ, CZ, DZ (hyperbola rotated about the Z-axis)
GNRLFIT	A, B, C, D, E, F (general fit rotated about the Z-axis)
CURVE TABLE NPT NN	(X1,Z1), (X2,Z2), ..., (XNN,ZNN) (curve table rotated about the Z-axis)

TABLE B33. GEOMETRY PARAMETER NAMES AND DEFAULTS

Equivalent parameter names	Default value (cm)	C array element
XO, X1, XL, XLEFT	-1.0E20	C(1)
XC, XCNTR, XCENT, XCENTER	0.0	C(1)
X2, XR, XRIGHT	1.0E20	C(4)
X3	-1.0E20	C(7)
X4	1.0E20	C(10)
YO, Y1, YL, YB, YBOT, YBOTTOM	-1.0E20	C(2)
YC, YCNTR, YCENT, YCENTER	0.0	C(2)
Y2, YR, YF, YT, YTOP	1.0E20	C(5)
Y3	1.0E20	C(8)
Y4	-1.0E20	C(11)
ZO, Z1, ZL, ZB, ZBOT, ZBOTTOM	-1.0E20	C(3)
ZC, ZCNTR, ZCENT, ZCENTER	0.0	C(3)
Z2, ZR, ZT, ZTOP	1.0E20	C(6)
Z3	-1.0E20	C(9)
Z4	1.0E20	C(12)
AY, A	0.0	C(10)
BY, B	1.0	C(11)
CY, C	0.0	C(12)
DY, D	1.0	C(13)
E	0.0	C(14)
F	0.0	C(15)
AZ	0.0	C(14)
BZ	1.0	C(15)
CZ	0.0	C(16)
DZ	1.0	C(17)
R, RAD, RADIUS	HOB*1.0	C(18)

TABLE B34. BURN2 PARAMETERS

Parameter	Function/default value
VDET	Detonation velocity 8.5E8 cm/s
TDET	Detonation time T0 (start time for the problem)
XDET	X-coordinate of the detonation point Note: If only XDET (or YDET) is specified, the detonation is assumed to be a line detonation from X=XDET (or Y=YDET). If neither is specified, both default to 0.0

TABLE B35. PARAMETERS TO SPECIFY LOCATION OF GFOR

Parameter	Function	Default value
XCC YCC ZCC	Specify coordinates of GFOR Origin with respect to the MFOR origin	0.0 cm 0.0 cm 0.0 cm
ANGLA	Rotation about the Z-axis of the GFOR	0.0 deg
DANA	Incremental angle	0.0 deg
NDA	Number of increments	0
ANGLB	Rotation about the Y-axis of the GFOR (3-D)	0.0 deg
DANB	Incremental angle	0.0 deg
NDB	Number of increments	0
ANGLC	Rotation about the X-axis of the GFOR (3-D)	0.0 deg
DANC	Incremental angle	0.0 deg
NDC	Number of increments	0

When using the PACKAGE option for filling the mesh, up to nine delete geometries may be specified. The following:

```
PACKAGE AIR RHO=1.2E-3 I=1.0E9 RECTANGLE
DELETE CIRCLE XC=0.0 YC=1.0E6 RAD=5.0E5
PACKAGE AIR RHO=1.2E-3 I=1.0E11 CIRCLE XC=0.0 YC=1.0E6 RAD=5.0E5
```

causes a sphere of hot air with a radius of 5.0e5 cm to be inserted at X=0.0, Y=1.0e6 cm. The word DELETE is optional: a geometry name not preceded by PACKAGE material is assumed to be a delete geometry.

The material can be specified to be packaged INSIDE (default) or OUTSIDE the geometry. For open geometries, INSIDE refers to above the curve (or surface) in the geometry frame of reference.

Many geometry parameter names are equivalent and all have default values as listed in Table B33. Values read from the KEEL input file are stored in the element of the coefficient array (C) corresponding to the parameter name preceding the value.

For the option BURN=2, the parameters listed in Table B34 are available to define the high explosive detonation properties.

Each package geometry is defined with respect to the geometry frame of reference (GFOR). The GFOR defaults to the mesh frame of reference (MFOR). For more flexibility, the GFOR may be relocated with respect to the MFOR, using the parameters listed in Table B35.

A typical package set is structured

```
PACKAGE MAT1 GEOM1 A=a B=b ...
XCC=x YCC=y ZCC=z ANGLA=aa ANGLB=bb ANGLC=cc
DANA=da DANB=db DANC=dc NDA=na NDB=nb NDC=nc
DELETE GEOM2 A=a B=b ...
XCC=x YCC=y ZCC=z ANGLA=aa ANGLB=bb ANGLC=cc
DANA=da DANB=db DANC=dc NDA=na NDB=nb NDC=nc
```

where the parameters listed after the geometry name can be in any order, but the parameters relating the location of the GFOR to that of the MFOR must occur after the geometry parameters. Parameters specifying the location of the GFOR with respect to the MFOR refer only to that particular geometry, and do not carry through to subsequent PACKAGE or DELETE geometries.

Subroutine INIT determines if a particular subcell contains the material being packaged, by calculating the coordinates of the center point of the subcell (given with respect to the MFOR) with respect to the GFOR. If the coordinates specify an interior point for the PACKAGE geometry and an exterior point for all DELETE geometries, the volume of this subcell is "filled" with the specified material density and energy. The momenta associated with the mass added will also be accumulated for this cell (if the packaged material has velocity components specified). Velocity components are always stated with respect to the MFOR.

Stations are generated by specifying the coordinates of each station, or group of stations. These coordinates may be Eulerian, Lagrangian, or mixed. A station with one Lagrangian coordinate is free to move in that direction.

The KEEL input file is terminated by the word END. If END is omitted, the file is read until an end-of-file is encountered.

2. HULL

The ZBLOCK options listed in Table B4 (with H or H* under the program heading) may appear in a HULL input file, after the keyword INPUT. Although the parameters with H* under the program heading may be changed in HULL input, the user is cautioned that unless the option was originally specified in the KEEL run, it cannot be "turned on" in HULL. Doing so implies that more variables per cell are available than what KEEL established. Such an option, however, may be set to zero in a HULL run and then reset to its original value on a subsequent run.

A HULL input file should conform to the format

HULL PROB xxxxx.xxxx
(C or CYCLE=cycle)
(T or TIME=time)
(INPUT parameters and parameter/value pairs)

where items in parentheses are optional. The parameters and parameter/value pairs may be ZBLOCK options or parameters listed in Table B36.

TABLE B36. BASIC HULL PARAMETERS

Parameter	Function/value
HULL	Identifies HULL input file
PROB or PROBLEM	Next word is the problem number
C or CYCLE	Before INPUT, identifies desired restart cycle for BOW
T or TIME	Before INPUT, identifies desired restart time for BOW
INPUT	*Terminates BOW section of input*
TIMES	Option to determine when restart files will be created 1. Standard times (default) 2. Given times (from array GTIME) 3. At specified time intervals (DMPINT)
DMPINT	Time (s) between restart files
DCYCST	Delta cycle stop (number of cycles allowed for this run)
CSTOP	Total number of cycles allowed for this problem
MRELER	Max relative error allowed (mass or energy)
RTSTOP	Max run time for this run (h)
VECTOR	If found by PLANK, causes vector code to be produced
NOVECT	If found by PLANK, causes scalar code to be produced (default)

As explained in Section B-I1, HULL periodically (every ICNTRL-th cycle) produces an ASCII file called HULLSTATUS which contains the information listed in Table B6. Program HULL stores HULLSTATUS under the problem subdirectory, so that a user can get a copy of it to monitor the problem's progress.

Also as previously explained, HULL looks for file HULLCNTRL before creating HULLSTATUS, to allow a user to change various operating parameters. The lines listed in Table B7 may also be included in the HULL input file after the keyword INPUT, if the user wishes to set these parameters at the beginning of the run. The functions of these adjustable parameters are explained below.

a. Autopriority--This parameter, if set, causes the problem priority to automatically be reduced to standby during prime shift (0600 hours through 1800 hours on weekdays), to take advantage of reduced computer rates. During all other shifts, the priority is raised to the load priority (default 1.0). To accomplish the automatic priority changes, subroutine CNTROL, if given the start day of the week (Mon, Tue, etc.), prevents the priority from being reduced on weekends.

b. Priority--This parameter sets the load priority to whatever valid value is desired. If AUTOPRIORITY is off, the priority is set to the load priority value. If AUTOPRIORITY is on, the action taken depends on the day of the week and time, as explained above.

c. STABF--This parameter is also a ZBLOCK variable. Valid values are between 0.1 and 0.9. STABF is the stability factor which determines the time step from the Courant condition.

d. Timelimit runtime--This parameter changes the job run time, either by reducing the originally requested run time, or by increasing a previously reduced time back to the original remaining time.

e. Timelimit stoptime--This parameter can be used to establish or cancel a desired date/time when the problem must be stopped. When turning STOPTIME on, the user specifies the month, day, hour, and minute when the problem is to be terminated.

f. Control cycle or icntrl-- These two parameters establish a new value for ICNTRL (default 10), which is the number of cycles between creation of new HULLSTATUS files.

g. DMPINT--This parameter sets the TIMES option to three (Table B36) and sets the time interval between restart files to the value (in seconds) specified.

h. End--Terminates the input file.

3. PULL

A PULL input file (Table B37) should conform to the format

PULL PROB xxxxx.xxxx
Control commands
Global plot commands
(OVERLAY)
Plot commands
(ENDOVERLAY)
Plot commands

where values in parentheses are optional. Example PULL input files appear at the end of this section.

TABLE B37. PULL CONTROL COMMANDS

Parameter	Function
PULL	Identifies PULL input file
PROB or PROBLEM	Next word is the problem number
FTIME	First time to be plotted (default, first restart file)
LTIME	Last time to be plotted (default, last restart file)
DTIME	Time increment between files to be plotted
THETA	Rotation angle for plots (default, 0.0 deg)
FACTPL	Plot factor used in subroutine UPLOTI (default, 1.0)
BD	Defines size of the border (default, 1.2 in)
XL	Defines width of plots (default, 10.0 in)
YL	Defines height of plots (default, 10.0 in)
MOVIE	Not implemented
SIZE=LL.HH	Instructs PULL to put LL plots across the defined plotting area and HH plots down, making an LL by HH array of plots on one plot frame (default, 1.1)
CTIME	Selects specific restart times to be plotted
CCYCLE	Selects specific restart cycles to be plotted
DEVICE	Selects a device for multiple device plot packages
STIME	Requests plots for standard times
SLIDES	Inserts blank frame between plots
NORMAL	Instructs PULL to draw plots in order specified

The PULL plot commands are of the form

DDDCCCCPPPP (parameters)

where DDD is a HULL data name (always required), CCCC is an optional calculation quantity, and PPPP is the type of plot requested. The parameters enclosed in parentheses define individual plot commands that may differ from the previously defined global plot commands (Table B38). Each parenthesis must be isolated by one or more blanks. Examples of plot commands are

DCONT -- Density contours
PVHST -- Pressure vertical histogram
S2ICONT -- Second stress invariant contours

TABLE B38. PULL GLOBAL PLOT COMMANDS

Parameter	Function
XMIN	Min X-coordinate to be plotted
YMIN	Min Y-coordinate to be plotted
XMAX	Max X-coordinate to be plotted
YMAX	Max Y-coordinate to be plotted
DX	X-Plot scale increment (default, scale from XMIN to XMAX)
DY	Y-Plot scale increment (default, scale from YMIN to YMAX)
CFACTOR	Scales plot label from 0.5 to 2.0 (default, 1.0)
ZMIN ZMAX	Define range of data to be plotted in 3-D plots (default, entire range of data)
CONTV	Defines contour values for contour plots (default, calculate contours from the data)
COLORMAP	Indicates color map for Tektronix 4100 terminals; up to 16 colors can be specified: C11,C21,C31,... Color indices C12,C22,C32,... Hues C13,C23,C33,... Saturations C14,C24,C34,... Brightnesses (default is terminal default color map)

TABLE B38. CONTINUED.

Parameter	Function
HIST=H1,H2,H3	Selects rows (or columns) for histogram plots (first coordinate, last coordinate, increment)
XPLANES YPLANES ZPLANES	Specifies planes to be plotted on 2-D plots from a 3-D problem. XPLANES=X1,X2,X3 indicates planes X=X1 through X=X2 incremented by X3.
EYE=EX,EY,EZ	Selects eye position for 3-D plots (default, appropriate position determined by PULL)
ICTRS	Number of contours to be calculated
MAT	List of material numbers (or names) for which plots are desired
COORD	Print Max and Min values on the plot
NOCOORD	Do not print Max and Min values on the plot (default)
CURV	Nonlinear interpolation contour plots (default)
NOCURV	Linear interpolation on contour plots
XLOG YLOG	Plot logarithmic data in the X (or Y) direction for histograms
NOXLOG NOYLOG	Plot linear data in the X (or Y) direction for histograms (default)
REF	Instructs PULL to reflect the contour and vector plots at the axis (or plane) of symmetry
NOREF	No reflection of data (default)
CGS	Centimeter/gram/second system of units (default)
ENGLISH	English system of units
GRID	Draw grid lines on plots
NOGRID	Do not draw grid lines on plots (default)
LOGD	Contour values incremented logarithmically
NOLOGD	Contour values incremented linearly (default)
LOGV	Vector plots use logarithmic scale
NOLOGV	Vector plots use linear scale (default)
CELLS	Put cell numbers on the plot (default)
NOCELLS	Do not put cell numbers on the plot
LEGEND	Include plot legend (default)

TABLE B38. CONCLUDED.

Parameter	Function
NOLEGEND	Suppresses the legend
NUMCONT	Put contour numbers on the plot (default)
NOCONTN	Suppress the numbers (default for color plots)
COLOR	Selects color plotting
DEFINECOLOR	Color parameters--color name, number, and intensity
ENDCOLOR	
NOCOLOR	Suppresses color plotting (default)
ADJCOORD	Adjusts scaling increments to 1.5 and 2.5 (default)
NOADJUST	Suppresses the adjustment
ARATIO	Insures aspect ratio (DX/DY) is one (default)
NORATIO	Allows each axis to be scaled independently

Available data names are listed in Table B39. The various plot types are listed in Table B40. The user must define workable combinations of these parameters. The two calculation quantities which exist are

GRAD -- Calculate gradient of a scalar
CURL -- Calculate curl of a vector

TABLE B39. PULL DATA NAMES

Name	Quantity
D	Material density
RD	Relative density
OD	Over density
P	Hydrostatic pressure
RP	Relative pressure
OP	Over pressure
E	Specific internal energy
RE	Relative internal energy
OE	Over energy
U	X-component of velocity
V	Y-component of velocity
W	Z-component of velocity
K	Specific kinetic energy
TE	Specific total energy
SRR	Radial stress component (2-D)
SZZ	Axial stress component (2-D) (or Z-component 3-D)
SHH	Hoop stress component (2-D)
SXX	X-stress component (3-D)
SYY	Y-stress component (3-D)
SRZ	Shear stress (2-D)
SXY	Shear stress (3-D)
SXZ	Shear stress (3-D)
SYZ	Shear stress (3-D)
S2I	Second stress invariant
SP1	Max principal stress
SP2	Min principal stress
ERR	Radial strain component (2-D)
EZZ	Axial strain component (2-D) (or Z-component 3-D)
EXX	X-strain component (3-D)
EYY	Y-strain component (3-D)
ERZ	Shear strain (2-D)
EXY	Shear strain (3-D)
EXZ	Shear strain (3-D)

TABLE B39. CONCLUDED.

Name	Quantity
EYZ	Shear strain (3-D)
E2I	Second strain invariant
EP1	Max principal strain
EP2	Min principal strain
TEMP	Material temperature
ERAD	Radiation energy density
VEL	Velocity vector
SV	Stress vector
HV	Strain vector

TABLE B40. PULL PLOT-TYPE NAMES

Name	Plot type
HHST	Horizontal histogram
VHST	Vertical histogram
CONT	2-D contour
RAST	Raster scan color plot of 2-D data (not fully implemented)
SURF	3-D contour plot of 2-D data
PERS	3-D perspective plot
VECT	Vector plot
FILL	Fill plot (not fully implemented)
PARTICLE	Special request (no other parts to it) to plot particles

PULL allows users to construct overlay plots (plots in which two or more data, or plot, types are drawn on the same frame). Different types may be superimposed, or they may be drawn on separate halves of the frame. An example of a superimposed overlay is velocity vectors, plotted over pressure contours. A split representation example is a plot of pressure contours on the left, with a pressure vertical histogram on the right half of the frame. Table B41 contains individual plot parameters.

TABLE B41. INDIVIDUAL PLOT PARAMETERS

Parameter	Function/value
OVERLAY	Indicates the start of a set of plots to be placed on one frame
( )	Plot parameters for this particular set of plots that differ from the global parameters must be enclosed in parentheses (Table B38)
SPLIT	Designates that overlaid plots will be split
LEFT	Puts plot on left half of split frame
RIGHT	Puts plot on right half of split frame (default)
LABEL	Defines top label as $string$ (default, data name)
XLAB	Defines replacement X-axis label as $string$
YLAB	Defines replacement Y-axis label as $string$ ($ may be any character not in the string)
ENDOVERLAY	Terminates this overlay plot set

a. Example PULL input files--The following example PULL input files and descriptions provide examples for user reference.

The following input file

```
PULL PROB 10.66 COORD DCONT PCONT
VVHST CTIME 1.0E-6 2.0E-5 5.0E-5
```

causes density contours and axial-velocity vertical histograms to be produced for problem 10.66 at times 1.0, 20.0, and 50.0 μs. Coordinates for the minimum and maximum values in the contour plots are printed on the plots produced.

The following input file

```
PULL PROB 19.84
OVERLAY ( REF ) PCONT PARTICLES
ENDOVERLAY
```

produces plots of pressure contours with particle positions plotted on the same frame, for all times found for problem number 19.84. The resultant plot (the left half of which is a mirror image of right half) is reflected about the axis of symmetry.

The following input file

```
PULL PROB 20.01
OVERLAY ( SPLIT )
DCONT 1.0 2.0 5.0 7.0 10.0 ( LEFT )
PCONT ( RIGHT )
PARTICLES ( RIGHT )
ENDOVERLAY
S2ICONT
```

specifies a composite overlay plot. The left half of that plot consists of density contours (contour values supplied); the right half is made up of pressure contours, with particle positions plotted over these contours. A separate frame containing the plot of second stress invariant contours, is also produced.

The following input file

```
PULL PROB 4.0
COLOR TEKCOLOR=1
S2IRAST
OVERLAY ( SPLIT )
DRAST ( LEFT )
PRAST ( RIGHT )
ENDOVERLAY
```

specifies color (raster scan) contour plots for the second stress invariant, and composite overlay plots with density on the left and pressure on the right. Scaling may be reduced, if the plot does not fit the screen (Tektronix 4100 series terminal).

4. STATION.

A typical STATION input file is structured

```
PULL STATION PROB xxxxx.xxxx (TMIN=t1 TMAX=t2)
STATION (or STATIONS ) (TMIN=t1 TMAX=t2)
Station numbers (or ALL)
Plot names (or ALL)  (see Table B42)
END
STATION (or STATIONS)
Station numbers (or ALL)
Different plot names (or ALL)
END
"
"
"
```

where station numbers may be specified by "1,2,3,4,5" or "1 to 5" or "ALL". Different plots can be requested for different stations. Example STATION input files appear at the end of this section. Plot names are listed in Table B42.

TMIN and TMAX specified before the first STATION/END set will apply to all sets. If TMIN and TMAX are specified after the word STATION, they will apply only to that set. TMIN and TMAX default to all times available on the station file.

a. Example STATION input files--The following example STATION input files and descriptions provide examples for user reference.

The input file

```
PULL STATION PROB 10.6  TMIN=15.0E-6  TMAX=30.0E-6
STATION  10,11,12
PRESS DENS INTENGY  END
STATION 1 TO 9
OPRESS
END
```

TABLE B42. STATION PLOT NAMES

Name	Plot produced
TXX	X-stress component
TYY	Y-stress component
TZZ	Hoop stress (2-D) (Z-stress component 3-D)
PRESS	Pressure
OPRESS	Over pressure
YIELD	Equivalent flow stress
TXY	Shear stress
TXZ	Shear stress
TYZ	Shear stress
EXX	X-strain
EYY	Y-strain
EZZ	Hoop strain (2-D) (Z-strain 3-D)
EMAX	Max principal strain
EXY	Shear strain
EXZ	Shear strain (3-D)
EYZ	Shear strain (3-D)
U	X-velocity
V	Y-velocity
W	Z-velocity (3-D)
TOTVEL	Vector sum of velocity components
DENS	Density
IMP	Pressure impulse
INTENGY	Internal energy
STRESS	All components of stress on one frame
STRAIN	All components of strain on one frame
AX	X-acceleration
AY	Y-acceleration
AZ	Z-acceleration

produces plots of pressure, density, and internal energy, versus time, for stations 10, 11, and 12 (from 15 μs to 30 μs), and plots of overpressure versus time for stations 1, 2, 3, 4, 5, 6, 7, 8, and 9 (from 15 μs to 30 μs).

The input file

```
PULL STATION PROB 11.777
STATION ALL ALL END
```

produces plots for all variables versus time, for all stations. The produced plots cover the entire problem time.

The input file

```
PULL STATION PROB 12.11
STATION TMIN 0.0 TMAX 15.0e-6
1,2,4  ALL END
STATION TMIN 15.0e-6 TMAX 100.0e-6
5,6 ALL END
```

produces plots for all variables versus time for stations 1, 2, and 4 from 0 us to 15 μs, and plots for all variables versus time for stations 5 and 6 from 15 μs to 100 μs.

B-IV. GLOSSARY OF MAJOR HULL VARIABLES

The following variable names used in the HULL code are not listed in alphabetical order. Instead, they are listed in the order they exist as elements of common blocks, or in order of relative importance as arbitrarily determined by the author.

1. PROC HULLCOM

This PROC is used by most subroutines in HULL, to assure that the common blocks are identical. Common blocks in this PROC contain the coordinate arrays, the mesh array, the ZBLOCK, and several other very important variables. Table B43 lists the variables unique to the two-dimensional code; Table B44 lists the variable for the three-dimensional code; and Table B45 lists the variables common to both.

TABLE B43. VARIABLES IN PROC HULLCOM (2-D)

Common block	Variable	Function
COORD	X(I)	X-coordinate for column I
	Y(J)	Y-coordinate for row J
	DX(I)	X(I) - X(I-1)
	DY(J)	Y(J) - Y(J-1)
	TAU(I)	Area of column I
	RC(I)	Distance from axis to center of column I
REZVU	UREZV(I)	Rezone velocity for column I
	VREZV(J)	Rezone velocity for row J
GRAVY	GA(J)	Gravitational potential for row J (center)
	GP(J)	Gravitational potential for row J (top)
INDEX	J	Current row
TSTEP	DTMAX	Max inverse time for whole mesh
	CSMAX	Max sound speed in mesh
	VMAX	Max velocity in mesh
	ICS,JCS	Coordinates of CSMAX
	IVMAX,JVMAX	Coordinates of VMAX
	IDT,JDT	Coordinates of DTMAX
	TMAX	Max temperature in mesh
	ITM,JTM	Coordinates of TMAX
	CUTDT	
	DTRAD	Radiation time step
	IRT,JRT	Coordinates of cell determining DTRAD
	DTHYD	

TABLE B44. VARIABLES IN PROC HULLCOM (3-D)

Common block	Variable	Function
COORD	X(I)	X-coordinate of column I
	Y(J)	Y-coordinate of row J
	Z(K)	Z-coordinate of plane K
	DX(I)	X(I) - X(I-1)
	DY(J)	Y(J) - Y(J-1)
	DZ(K)	Z(K) - Z(K-1)
REZVUZ	UREZV(I)	Rezone velocity for column I
	VREZV(J)	Rezone velocity for row J
	WREZV(K)	Rezone velocity for plane K
GRAVY	G1(K)	Gravitational potential for plane K (center)
	G2(K)	Gravitational potential for plane K (top)
INDEX	K	Current plane
TSTEP	DTMAX	Max inverse time
	CSMAX	Max sound speed
	VMAX	Max velocity
	ICS,JCS,KCS	Indices of cell containing CSMAX
	IVMAX,JVMAX,KVMAX	Indices of cell containing VMAX
	IDT,JDT,KDT	Indices of cell containing DTMAX
	TMAX	Max temperature
	ITM,JTM,KTM	Indices of cell containing TMAX

TABLE B45. VARIABLES IN PROC HULLCOM (FOR 2-D AND 3-D)

Common block	Variable	Function
ZBLOCK	See Table B4	
ZBLK	ZBLK	Same as ZBLOCK
CONST	PI	3.14159.... (or 0.5 for DIMEN2 GEOM1)
	GAMMA	Gamma for air
(Blank common)	CSTOP	Cycle stop
	CTAPE	
	DCYCST	Delta cycle stop
	DEBUG	Debug prints
	DMPINT	Dump interval for TIMES=3
	DRADLOS	
	DTLD	Time step last dump
	DUMPN	Flag to indicate dump request
	FCYC	
	FPTIME	
	FT	
	ICM	Column containing center of mass
	IMAXM1	IMAX-1
	IMAXM2	IMAX-2
	IPD	
	IPMX	
	IRES	
	ISTAPE	
	JCM	Row containing center of mass
	JMAXM1	JMAX-1
	JPMX	
	KMAXM1	KMAX-1
	MRELER	Max relative error (for mass or energy)
	NA	Index for row (or plane) above
	NB	Index for row (or plane) below
	NBLKS	Number of data blocks containing mesh on TAPE4
	NDUMP	

TABLE B45. CONTINUED.

Common block	Variable	Function
	NHEC	Number of hydro variables in whole mesh
	NLTAPE	
	NN	
	NPIC	Number of particles in core
	NPLR	Number of particles in last record
	NPPR	Number of particles per plane
	NPPR	Number of particles per record
	NPREC	Number of particle records
	NPLIC	Number of planes in core
	NROWIC	Number of rows in core
	NSTAPE	
	NSLVPC	Number of slice variables per cell
	NSLVPP	Number of slice variables per plane
	NSLVPR	Number of slice variables per row
	NTAPE	
	NVARPB	Number of variables per block
	NVARPP	Number of variables per plane
	NVARPR	Number of variables per row
	PCHAN	
	PMAX	Max pressure in mesh
	PRINTL	
	PRINTW	Characters per line
	RADRAT	Radiation rate
	RELERR	Relative energy error
	REZONA	Flag for rezone, aft
	REZONB	Flag for rezone, below (or bottom)
	REZONF	Flag for rezone, fore
	REZONL	Flag for rezone, left
	REZONR	Flag for rezone, right
	REZONT	Flag for rezone, top
	RTSTOP	Run time stop
	SREAD	Flag to indicate successful read
	TBD	

TABLE B45. CONCLUDED.

Common block	Variable	Function
	TERMIN	Flag to indicate termination
	TIMES	Dump option
	TIMINC	
	TLD	Time last dump
	TMAGE	
	VMIN	Min significant velocity
	WHIZ	Efficiency measure (CPU time/[cell*cycle])
	MSG	Message array for terminal remarks
	RES	
	SRES	
HIC	H(N)	Hydro array

To simplify addressing hydro variables in the H array, the following system of equivalenced names is used in the HYDRO and FLUX routines. All variables for a particular cell are in sequence. The same variable for two adjacent cells in the same row are NH words apart.

Refer to Table B45 and assume the number of materials (NM) is two, and all possible options are desired.

TABLE B46. HYDRO ARRAY EQUIVALENCES FOR 2-D

H array	Equivalenced arrays		Value for first cell	Option
H(1)		P(0)	Pressure	
H(2)		U(0)	X-velocity	
H(3)		V(0)	Y-velocity	
H(4)		XI(0)	Internal specific energy	
H(5)		XM(0)	Total mass	
H(6)		XM(1)	Mass of material 1	
H(7)		XM(2)	Mass of material 2	
H(8)		XV(1)	Volume of material 1	
H(9)		XV(2)	Volume of material 2	
H(10)		XII(1)	Total internal energy for material 1	
H(11)		XII(2)	Total internal energy for material 2	
H(12)	HIST(1)	SRR(0)	Radial stress deviator	STRESS
H(13)	HIST(2)	SZZ(0)	Axial stress deviator	STRESS
H(14)	HIST(3)	STT(0)	Hoop stress deviator	ANISTRP
H(15)	HIST(4)	SRZ(0)	Shear stress deviator	STRESS
H(16)	HIST(5)	SRT(0)	Shear stress deviator (3-D)	SPIN/STRESS
H(17)	HIST(6)	SZT(0)	Shear stress deviator (3-D)	SPIN/STRESS
H(18)	HIST(7)	ANGMOM(0)	Angular momentum	SPIN
H(19)	HIST(8)	ERRH(0)	Radial strain	STRAIN
H(20)	HIST(9)	EZZH(0)	Axial strain	STRAIN
H(21)	HIST(10)	ERZH(0)	Shear strain	STRAIN
H(22)	HIST(11)	ERTH(0)	Shear strain	SPIN/STRAIN
H(23)	HIST(12)	EZTH(0)	Shear strain	SPIN/STRAIN
H(24)	HIST(13)	EPLASH(0)	Plastic strain	WORK
H(25)	HIST(14)	UMAXH(0)	Max compression	CRUSH
H(26)	HIST(15)	FMX(0)	Magnetic field X-component	MAGFLD

TABLE B46. CONCLUDED.

H array	Equivalenced arrays		Value for first cell	Option
H(27)	HIST(16)	FMY(0)	Magnetic field Y-component	MAGFLD
H(28)	HIST(17)	ERAD(0)	Radiation energy	RAD
H(29)		GETIME(0)	Detonation time	BURN2
H(30)		CV(0)	Radiation specific heat	RAD
H(31)		DLTA(0)	Change in energy	RAD
H(32)		TK(0)	Radiation temperature	RAD
H(33)		FYJ(0)	Back substitution coefficient	RAD

For the situation in Table B46, the number of hydro variables per cell (NH) is 33, so pressures for cells 1, 2, and 3 are in P(0), P(NH), and P(2*NH) or H(1), H(34), and H(67). Table B47 shows the equivalences for the 3-D code with NM = 2.

TABLE B47. HYDRO ARRAY EQUIVALENCES FOR 3-D

H Array	Equivalenced arrays		Value for first cell	Option
H(1)		P(0)	Pressure	
H(2)		U(0)	X-velocity	
H(3)		V(0)	Y-velocity	
H(4)		W(0)	Z-velocity	
H(5)		XI(0)	Specific internal energy	
H(6)		XM(0)	Total mass	
H(7)		XM(1)	Mass of material 1	
H(8)		XM(2)	Mass of material 2	
H(9)		XV(1)	Volume of material 1	
H(10)		XV(2)	Volume of material 2	
H(11)		XII(1)	Internal energy for material 1	
H(12)		XII(2)	Internal energy for material 1	
H(13)	HIST(1)	SXX(0)	X-stress deviator	STRESS
H(14)	HIST(2)	SYY(0)	Y-stress deviator	STRESS
H(15)	HIST(3)	SZZ(0)	Z-stress deviator	ANISTRP
H(16)	HIST(4)	SXY(0)	XY-shear stress	STRESS
H(17)	HIST(5)	SXZ(0)	XZ-shear stress	STRESS
H(18)	HIST(6)	SYZ(0)	YZ-shear stress	STRESS
H(19)	HIST(7)	EXXH(0)	X-strain	STRAIN
H(20)	HIST(8)	EYYH(0)	Y-strain	STRAIN
H(21)	HIST(9)	EXYH(0)	XY-shear strain	STRAIN
H(22)	HIST(10)	EXZH(0)	XZ-shear strain	STRAIN
H(23)	HIST(11)	EYZH(0)	YZ-shear strain	STRAIN
H(24)	HIST(12)	EPLASH(0)	Plastic strain	WORK
H(25)	HIST(13)	UMAXH(0)	Max compression	CRUSH
H(26)		DETIME(0)	Detonation time	BURN2

All variables showing equivalences to the HIST array are advected with the mass during the fluxing phase of the calculation. Those variables not showing HIST on the same line are not fluxed (the variable DETIME, and those following in 2-D).

2. HYDRO, FLUX, AND EOS COMMONS

The variables listed in Tables B48 and B49 are required only for the rows actually being processed.

TABLE B48. 2-D COMMON BLOCKS

Common block	Variable	Function	Option
SEOSET	RCSJ(I)	Column I Rho*C**2 (RCS)	
	CSJ(I)	Column I sound speed (CS)	
	AMUJ(I)	Column I bulk modulus	STRESS
	YLDJ(I)	Column I flow stress	STRESS
SHYDRO	PJ(I)	Column I boundary pressure	
	VJ(I)	Column I boundary velocity	
	PK(I)	Column I top boundary pressure	
	VK(I)	Column I top boundary velocity	
SSTRESS	USJ(I)	Column I boundary stress velocity	STRESS
	VSJ(I)	Column I boundary stress velocity	STRESS
	SRZJ(I)	Column I boundary shear stress	STRESS
	SRZZ(I)	Column I boundary axial stress	STRESS
SFLUX	FMJ(I)	Column I mass flux	NM1
	FEJ(I)	Column I energy flux	NM1
	UMOMJ(I)	Column I U momentum flux	NM1
	VMOMJ(I)	Column I V momentum flux	NM1
	HISTJ(N,I)	Column I HIST(N) flux	NHIST/NM1
	HISTL(N)	Left cell boundary HIST(N) flux	NHIST/NM1
	HISTR(N)	Right cell boundary HIST(N) flux	NHIST/NM1

Table B48. CONCLUDED.

Common block	Variable	Funtion	Option
	HISTB(N)	Bottom cell boundary HIST(N) flux	NHIST/NM1
	FVB(I)	Column I volume flux (or velocity)	FLUXER3
HOLDS	HS(NS)	Fractions of materials to be fluxed	FLUXER3
MFLUX	FMB(M,I)	Column I material M mass flux	FLUXER3
	FVB(M,I)	Column I material M volume flux	FLUXER3
	FIB(M,I)	Column I material M energy flux	FLUXER3
	FTIB(I)	Column I total energy flux	FLUXER3
	FUMOMB(I)	Column I U momentum flux	FLUXER3
	FVMOMB(I)	Column I V momentum flux	FLUXER3
	FHISTB(N,I)	Column I HIST(N) flux	FLUXER3

TABLE B49. 3-D COMMON BLOCKS

Common block	Variable	Function	Option
SEOSET	RCSK(I)	Column I Rho*C**2(RCS)	
	CSK(I)	Column I sound speed (CS)	
	AMUK(I)	Column I bulk modulus	STRESS
	YLDK(I)	Column I flow stress	STRESS
SHYDRO	PJ(I)	Column I front boundary Pressure	
	VJ(I)	Column I front boundary Velocity	
	PK(I)	Column I top boundary Pressure	
	WK(I)	Column I top boundary Velocity	
SSTRESS	USJ(I)	Column I	STRESS
	VSJ(I)	Front boundary	STRESS
	WSJ(I)	Stress velocities	STRESS
	USK(I)	Column I	STRESS
	VSK(I)	Top boundary	STRESS
	WSK(I)	Stress velocities	STRESS
	SYYJ(I)	Column I	STRESS
	SXYJ(I)	Front boundary Y-stress	STRESS
	SYZJ(I)	and shear stress deviators	STRESS
	SZZK(I)	Column I	STRESS
	SXZK(I)	Top boundary Z-stress	STRESS
	SYZK(I)	Shear stress deviators	STRESS
SFLUX	FVB(I,J)	Flux volume from Plane below	
HOLDS	HS(NS)	Fraction of materials to be fluxed	FLUXER3
MFLUX	FMB(M,I,J)	Mass, Volume, and internal energy being fluxed from plane	
	FVB(M,I,J)		
	FIB(M,I,J)	below for each material	FLUXER3

TABLE B49. 3-D COMMON BLOCKS. CONCLUDED.

Common block	Variable	Function	Option
	FTIB(I,J) FUMOMB(I,J) FVMOMB(I,J) FWMOMB(I,J)	Total energy and Momenta being fluxed from plane below	

3. VECTOR UNIQUE VARIABLES

The variables listed in Table B50 are used in the vectorized version of HULL.

TABLE B50. VECTOR VARIABLES

Block	Variable	Function	Option
ICDPTR	ICD(I)	Cell descriptor for cell I	
		= -1 island cell	
		= 0 fluid cell	
		= 2**im fluid cell containing im	
		= 99000 + Σ 2**im if material im is within the cell (multimaterial cell)	
	ISLND(J)	= 1 indicates some island cells within row J	ISLAND
		= 0 indicates no island cells within row J	
TSTEP	BURNTOT(J)	= 1.0 no unburned material within row J	BURN
		= 0.0 some unburned material within row J	
	NESOUT	Logical flag used to indicate how many cells are to be processed by subroutine eosset.	
		= true process only 1,...,iq+1	
		= false process 1,...,imax	

TABLE B5O. CONCLUDED.

Block	Variable	Function	Option
VTMPxx	VTMPxx(I)	Vector temporaries (xx=1,64) In general, local arrays are equivalenced to these vector temporaries, to conserve memory. Occasionally, they are used to pass values from one subroutine to another (calls from eossett or hydro to state).	
VECNMx	VECNMx(IM,I)	Two-dimensional vector temporaries (x=1,B) Used same way as VTMPxx, where local array requires two subscripts.	
EOSTMPx	EOSTMPx(I)	Vector temporaries Used only within EOS routines. Never used to pass values from one subroutine to another.	

END

DATE

FILMED

8-88

DTIC